BUILD A
RASPBERRY PI 5
MEDIA PLAYER

■ High-quality audio ■ Massive M.2 storage ■ Stream anything

£5.99

Raspberry Pi PRESS

9 772051 998001

Building an
RC plane display

Code a budget tracker
in Python

TOP 10
gaming accessories

HOW RASPBERRY PI BUILT A SILICON DESIGN TEAM
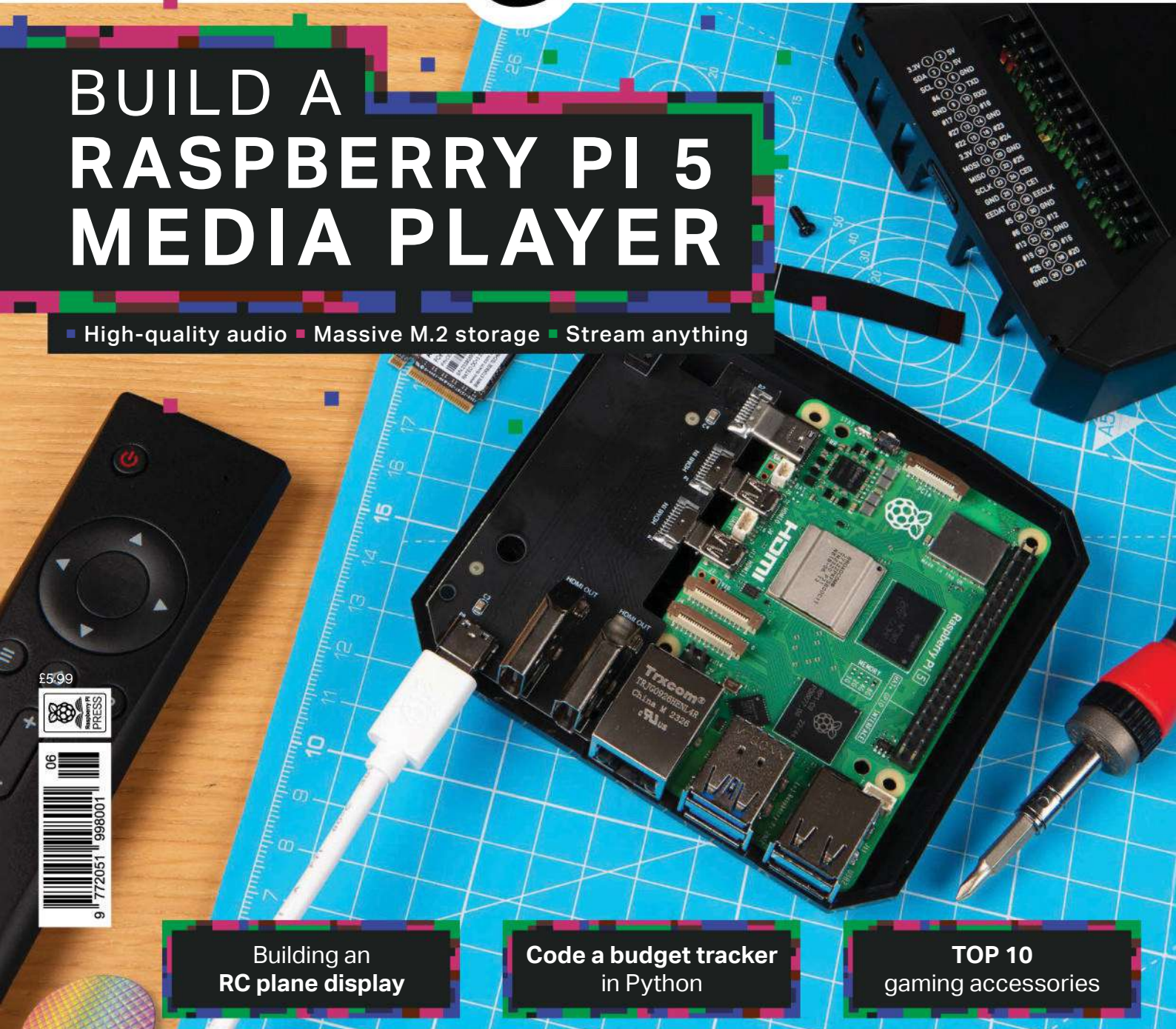
Industrial Raspberry Pi
ComfilePi

UL LISTED

Powered by **Raspberry Pi**

The ComfilePi is a touch panel PC designed with high-tolerant components and no moving parts for industrial applications. It features a water-resistant front panel, touchscreen, color LCD (available in various sizes), RS-232, RS-485, Ethernet, USB, I2C, SPI, digital IO, battery-backed RTC (real-time clock), and piezo buzzer.

Use the rear-panel 40-pin GPIO header to expand its features and capabilities with additional I/O boards. The ComfilePi is UL Listed and employs Raspberry Pi Compute Module.

Visit www.comfiletech.com

COMFILE TECHNOLOGY

WELCOME
to The MagPi 142

This month the sun is shining on a couple of new **Raspberry Pi productions.** We've got Raspberry Pi Connect (**page 12**) and new Compute Module 4 boards with up to 16GB of RAM (**page 14**).

Raspberry Pi 5 now has super-fast SSD storage, and PJ has celebrated by making an all-new media player (**page 36**). This one is much faster than its predecessor, and is perfect for video streaming, movie file playback, and high-quality audio. It truly is the best media player you can build.

Because the sun has got its hat on, we've got a range of summer projects to share with you (**page 72**). Everything from bird smart bird feeders, machine learning cameras and tide and weather trackers, to portable computer backpacks. There's even a rain detector that – with any luck – we won't need.

Have great fun with these projects. There's a lot going on in the world of Raspberry Pi, and we can't wait to see what you build with it all.

Lucy Hattersley Editor

GET A
RASPBERRY PI PICO W
WITH A SUBSCRIPTION!
PAGE **34**

Hello

Whether you're an

- **engineer**
- **designer**
- **purchaser**
- **maker**

we've got the products, services, and business solutions to help move you forward.

We're DigiKey

Explore and connect today.

digikey.co.uk

DigiKey

we get technical

Contents

> Issue 142 > June 2024

BUILD A RASPBERRY PI 5 MEDIA PLAYER
36

28
RC plane OSD

26
Cat TV

56

Motion control a robot arm

60

Rescue your old backups

The Big Feature

72

Summer projects

78

CrowVi display

84

What's Ken Making interview

WIN
1 OF 10 M.2 **HAT+**

94

How Raspberry Pi built
a silicon design team

We meet Raspberry Pi's ASIC design team
to talk about chip design. By **Lucy Hattersley**

**Tammy
Julyan**

**ASIC Engineering
Director**
Tammy has worked
with Raspberry Pi
for nine years and is
responsible for the
tiny Raspberry Pi
logo on the silicon.

**Nick
Francis**

**ASIC Technical
Director**
Nick has been with
Raspberry Pi for
seven years and
has worked on
RP1 and RP-2040.

Alongside building single-board
computers and microcontroller boards,
Raspberry Pi is heavily involved in silicon
design. For nearly ten years now Raspberry Pi
has been building an ASIC (application-specific
integrated circuit) team in Cambridge to design
and produce custom silicon chips for its products

This intricate process involves designing
integrated circuits on a silicon wafer. We sat down
with Tammy Julyan, ASIC Engineering Director and
Nick Francis, ASIC Technical Director to chat about
Raspberry Pi's ASIC team, and how they go about
designing silicon chips. It's a fascinating deep dive
into bare metal computing.

Raspberry Pi makes computers using
components like the Broadcom BCM2712
System-on-Chip, Renesas DA9091 power
management unit, and Infineon CYW43455
Bluetooth and Wi-Fi chip.

However, for the last 10 years Raspberry Pi has
also been designing its own chips, such as the
RP2040 microcontroller found in Raspberry Pi Pico
and the RP1 I/O controller found in Raspberry Pi 5.
These contain smaller blocks, often referred to as
IP (intellectual property) designed by Raspberry Pi
or bought in from elsewhere and integrated into
Raspberry Pi's chips.

Integrated circuits

Silicon is basically sand, which is the world's
most abundant element after oxygen. Silicon is
a semiconductor, which means it can conduct
electricity under some conditions but not others,
making it ideal for controlling electrical current. It
is turned into silicon wafers, which have patterns
etched onto them to create integrated circuits.

An integrated circuit (IC) also known as a
silicon chip, microchip, microelectronic circuit or
computer chip, is an integrated system of multiple
miniaturised and interconnected components
embedded into a thin substrate of semiconductor
material. ICs are made up of millions of tiny
transistors. They have many different functions,
and can be found in products ranging from
automotive, computing, and medical equipment.

Silicon design is specifying, designing and
testing an IC. It also involves creating the physical
layout of the chips' circuits. This consists of a
team of skilled engineers working together using
their expertise to ensure the final design meets the
original requirements.

The first step is the specification. Requirements
can come from several sources, an innovative
new idea, customers, internal product needs,
and marketing forecasts. For example: RP1 was
designed to hang off the main processor as a
southbridge, handling most input and output
capabilities for Raspberry Pi 5. It controls the
camera, display, Ethernet, USB, and GPIO (see
magpi.cc/rp1doc).

"We start with a team of people from across
the different groups including software, hardware
and management to discuss a new chip: 'let's
make X to do Y'," says Tammy. Architecting a
new chip is an iterative process, and the whole
team will meet multiple times to discuss what the
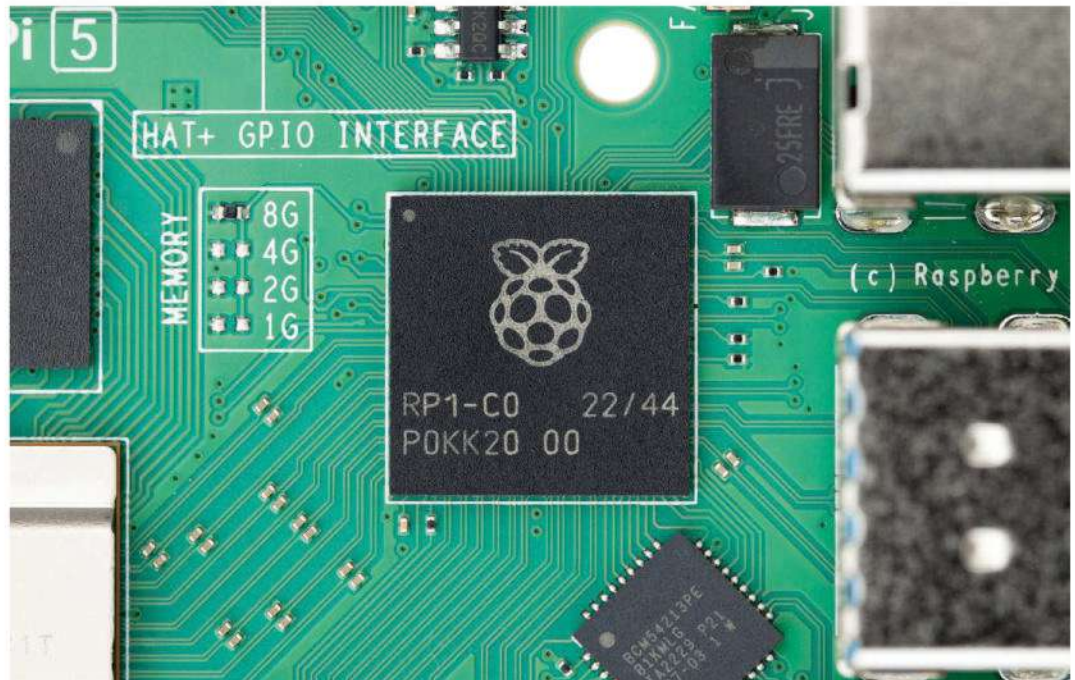
"shape and feel of the thing" needs to be. Many factors are considered: performance, timescales, make vs. buy, interfaces, features, power, board-level design, and software.

Most devices are system-on-chips (SoCs), they have a controlling processor, some memory, and some peripherals which perform functions such as data processing or an interface to the rest of the system. "You need to consider how these blocks communicate with each other and what software will run to control the system. The ASIC team also needs to consider a whole host of other things like how to start up the device, what clocks are running, how to cross clock domains. There's a

> ❝ Once you've got a list of requirements you start to build the reality ❞

very long list. So now you have an architecture and can start to build your chip."

Some of the blocks Raspberry Pi may already own, and can be reused from a previous design, while some the design team needs to build from scratch or buy. However buying in a block – or IP – isn't an easy process: they need to choose the right vendor, configure the block correctly, integrate and verify it.

"Chip design is written in a language called

Verilog" explains Nick. It is a bit like C but "it doesn't execute in order; it executes all at once". Verilog is a hardware description language (HDL), that describes in code how the circuit functions.

Testing the design

Once the design is complete you have to test that it works as expected before it is committed to manufacture, this activity is known as verification. Verification is a huge part of the silicon design process: "I think the biggest difference is that software – even if it's well tested – can be more easily changed afterwards," says Nick. So "most of my time is spent testing".

Software code, either C or Verilog, is written to stimulate the design and the output is checked. This testing can be at the block or chip level, normally done through simulation.

Some block or chip-level testing uses constrained random techniques to generate test conditions that are hard to predict. "We can run many of these random tests to help find corner case issues."

As well as simulation tools, FPGAs (field programmable gate arrays) are used to test the design. An FPGA is physical hardware that can be reconfigured to emulate different hardware circuits, this enables testing to take place in near-real-time in hardware (instead of being simulated in software, which is much slower, being limited by available processing power). FPGAs are useful

▶ An example IC layout created with a place and route tool

for system integration testing and software development before the chip is available.

Once the design is functioning as expected, the HDL needs to be converted into a physical design which is a representation of the geometric shapes that will create the tiny transistors and connections on the final chip.

Extra functionality is added to the design to support manufacturing tests to check for defects that could affect the correct operation of the IC. Some of these extra Design For Test (DFT) functions are structural and are inserted using tools which also generate test patterns to test logic cells, and some are internal Built In Self Test (BIST) engines to test things like memory. To convert the HDL, it is loaded into a synthesis tool along with a timing constraints file called SDC (standard

▼ A silicon wafer containing an array of integrated circuits. This is cut to produce the individual chips
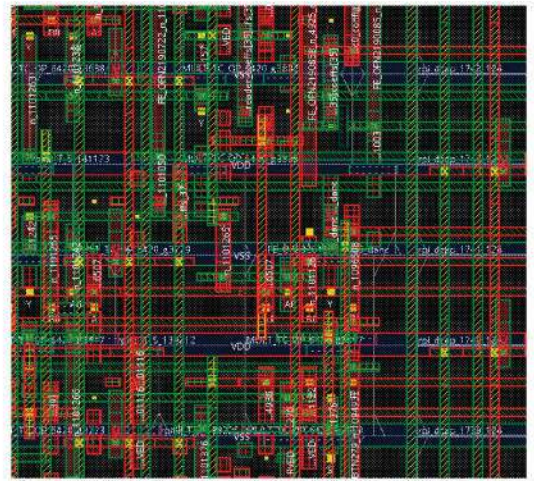
design constraints) and libraries: for example, logic gates, memories and analogue blocks. The synthesis tool converts the Verilog into a gate-level netlist which is functionally equivalent to the HDL description, making architectural decisions based on how fast the design needs to run, while also considering the power and area to get the most optimal design.

After synthesis, the gate-level Verilog netlist is loaded into a 'place and route' tool where a floorplan is created, positioning the memories, analogue blocks and any IO. A metal power grid is added to feed the gates with power across the floorplan. The standard cells are then placed in rows by the place and route tools, the metal wires connecting up the standard cells, memories, etc are routed. The tool continuously checks timing and when possible fixes any slow paths. The physical design is checked to make sure it is functionally correct, meets the timing requirements and can be manufactured correctly. When complete this is sent to the wafer foundry to be manufactured.

The "manufacturing process itself is interesting," says Tammy. The physical design is translated into a set of masks, the number of which increases as technology advances.
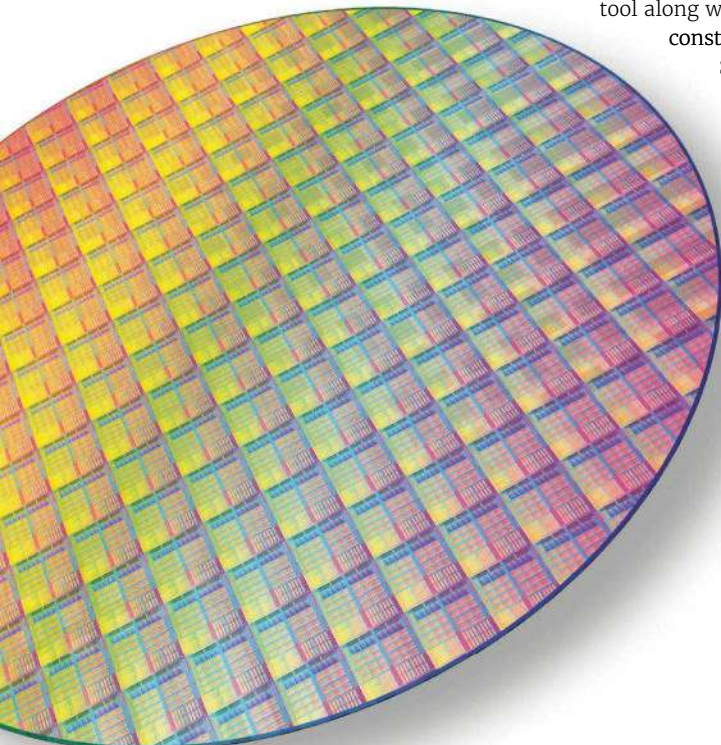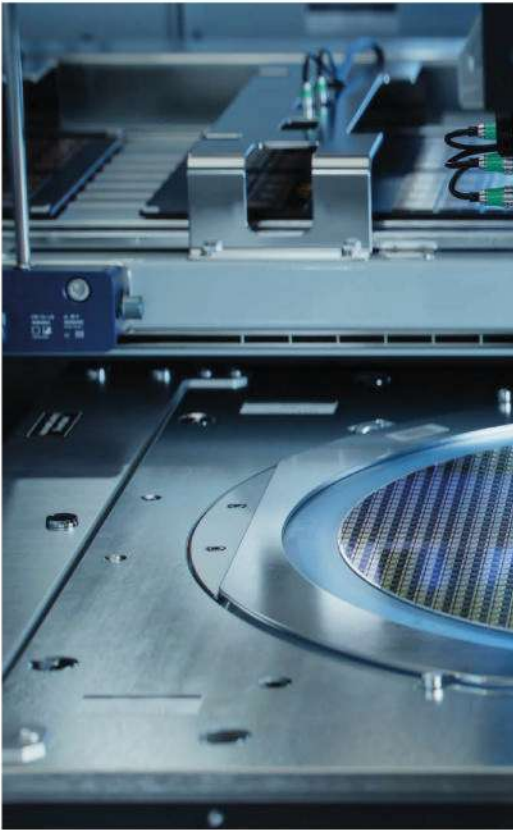
" Raspberry Pi has developed from the ground a highly talented design team "

They grow the silicon crystal by dipping a 'seed crystal' into molten silicon until it turns into a large cylindrical ingot. Then they are sliced thinly into wafers, ground and polished. Once the wafers are created they go through multiple process steps to create the IC. Photolithography is used to transfer the geometric patterns from the mask onto a photosensitive layer applied to the wafer, and in addition to oxide growth, etching, deposition, ion implantation and metalisation the IC is created. When complete, the wafer is covered in rows of ICs which need to be divided up into individual chips by sawing and placed in their packages. Once packaged they are tested to ensure nothing has gone wrong in the manufacturing process.

Ten years ago Raspberry Pi had no ASIC design capability, and since then it has developed from the ground a highly talented design team. We look forward to seeing what they create next. M

▲ A semiconductor Wafer after the dicing process. Silicon dies are being extracted by pick and Place Machine

Hear about RP1

Raspberry Pi's founder and CEO, Eben Upton and Chief Technology Officer James Adams discuss RP1 in a video on Raspberry Pi's RP1 documentation page. RP1's datasheet has detailed information about Raspberry Pi's RP1 silicon chip.

▶ **magpi.cc/rp1doc**

Introducing
Raspberry Pi Connect

Remote control Raspberry Pi from around the world.
By **Gordon Hollingworth**, Raspberry Pi CTO (Software)
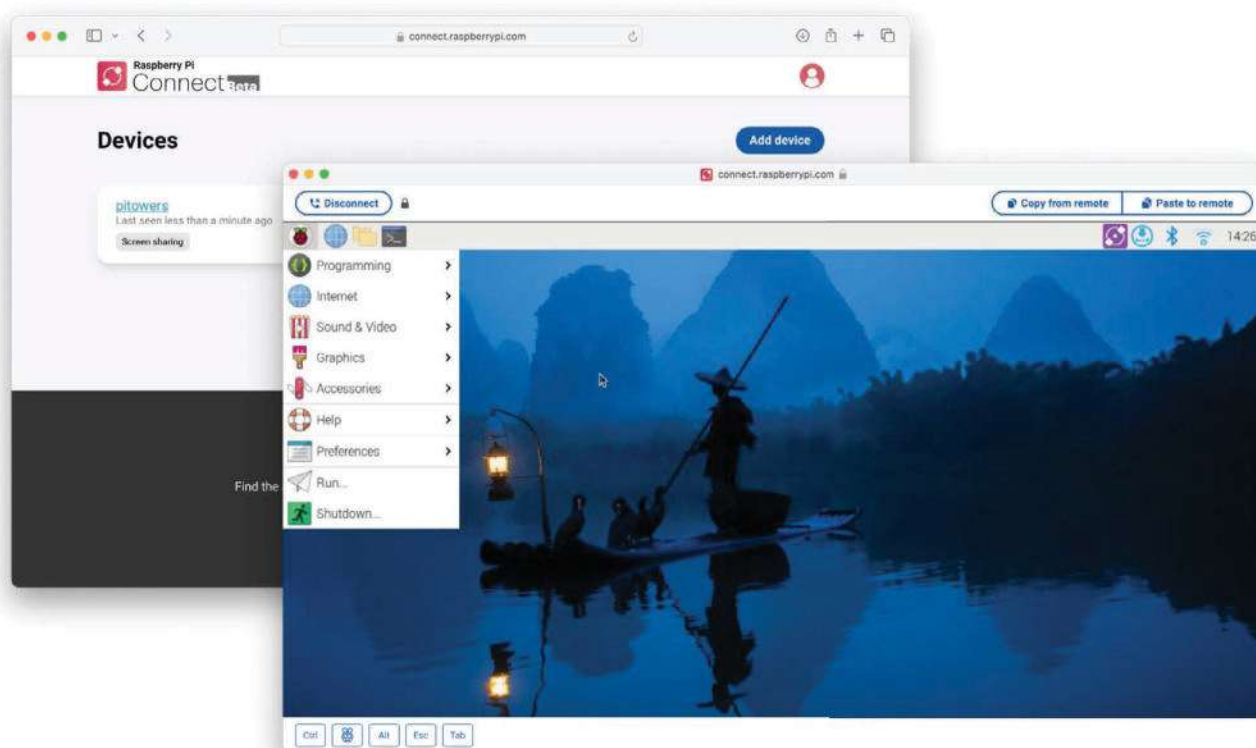
We're pleased to announce the beta release of Raspberry Pi Connect (magpi.cc/connect): a secure and easy-to-use way to access your Raspberry Pi remotely, from anywhere on the planet, using just a web browser.

It's often extremely useful to be able to access your Raspberry Pi's desktop remotely. There are a number of technologies which can be used to do this, including VNC, and of course the X protocol itself. But they can be hard to configure, particularly when you are attempting to access a machine on a different local network; and of course with the transition to Wayland in Raspberry Pi OS Bookworm, classic X remote desktop support is no longer available.

We wanted to be able to provide you with this functionality with our usual 'it just works' approach. Enter Raspberry Pi Connect.

How do I get Raspberry Pi Connect?

First of all, Raspberry Pi Connect needs your Raspberry Pi to be running a 64-bit distribution of Raspberry Pi OS Bookworm that uses the Wayland

window server. This in turn means that, for now, you'll need a Raspberry Pi 5, Raspberry Pi 4, or Raspberry Pi 400.

Assuming you're using one of these models, make sure you have the latest Raspberry Pi OS Bookworm from Raspberry Pi Imager, open a terminal, and enter:

```
sudo apt update
sudo apt upgrade
sudo apt install rpi-connect
```

Now reboot your Raspberry Pi, and you'll find a new icon in your system tray at the top right of your screen. Click this icon and choose 'Sign in' to get started. Hopefully you'll find the instructions easy enough to follow, but if you need it, there's extra documentation that covers known limitations during the beta.

What happens under the hood?

I asked Paul Mucur, who runs web development at Raspberry Pi, to explain how the underlying technology works:

"When you use Raspberry Pi Connect from a web browser to connect to your Raspberry Pi device, we establish a secure peer-to-peer connection between the two using WebRTC: the same real-time communication technology that underpins the in-browser clients for Zoom, Slack, Microsoft Teams, and Google Meet," he says.

"Our 'rpi-connect' daemon for Raspberry Pi OS is responsible for listening out for new screen sharing sessions from the Raspberry Pi Connect website, and negotiating the best possible, or lowest latency, connection between the in-browser VNC client and a VNC server running on your device. In general, once a connection is established, no traffic need pass through our servers.

> 🔴 Our intention is that Raspberry Pi Connect will remain free for individual users 🔴

"If for any reason it is not possible to establish a direct connection between your browser and Raspberry Pi device, rpi-connect and your browser may instead opt to securely relay traffic through our servers, encrypting it with DTLS."

Peer-to-peer and relayed connections

At the moment, the Raspberry Pi Connect service has just a single relay (TURN) server, located in the UK. This means that if rpi-connect chooses to relay traffic, the latency can be quite high. Hovering over the padlock icon in your browser while connected will reveal whether your connection is being relayed or not, so you can tell whether changes to your networking setup might improve connectivity.

Our intention is that Raspberry Pi Connect will remain free (as in beer) for individual users with non-relayed connections, with no limit on the number of devices. We don't yet know how many people will need to relay their traffic through our TURN servers; we'll keep an eye on the use of bandwidth and decide how to treat these connections in future.

As I said at the beginning, Raspberry Pi Connect is in beta at the moment, so please bear in mind that you might come across the occasional limitation or imperfection. We think lots of people will find it useful, and we hope you like the sound of it enough to follow the instructions above or in the Connect documentation to install it and try it out. You can let us know what you think in the Raspberry Pi Connect section of our forums (**magpi.cc/forum**). 🔴

Industrial Compute Module 4 upgraded!

New memory variants for the Raspberry Pi Compute Module family are ready for industrial applications. By **Dave Lee**

◀ Raspberry Pi Compute Module 4S is an industrial computer built around Raspberry Pi 4 technology with up to 16GB of onboard eMMC storage

Since 2014, we have provided the power of Raspberry Pi in a flexible form factor designed specifically for industrial and embedded applications, and we've been surprised and delighted to see the incredible variety of ways in which industrial customers use Raspberry Pi Compute Modules. We are pleased to announce that we have expanded our Compute Module 4S offering (**magpi.cc/cm4s**): these industrial boards are now available with additional SDRAM, and as well as the original 1GB variant, you can now choose from 2GB, 4GB, and 8GB options.

Compute Module 4S is based on Raspberry Pi 4's architecture. We designed it for industrial customers who are migrating

from Compute Module 3 or Compute Module 3+ (**magpi.cc/cm3plus**), and who are looking to retain the same form factor but would like greater computing power and more memory.

We will keep Compute Module 4S in production until at least January 2034, and we have produced a transition document specially to help users migrate to it from Compute Module 1, Compute Module 3, or Compute Module 3+. A full product brief (**magpi.cc/cm4sdoc**) and datasheet (**magpi.cc/cm4sdata**) are also available.

Compute Module 4S boards are in stock and available now from our Approved Resellers for industry, with a maximum lead time of six weeks.

> 🔴 **Industrial customers have used these boards in everything from self-pour beer taps to electric vehicle charging stations** 🔴

These modules are supplied only in bulk 200-unit boxes, with prices per unit starting from $25 in the US. Find your local Raspberry Pi Approved Reseller for industry (**magpi.cc/resellers**) and contact them directly to discuss sales.

What is Compute Module 4S used for?

Our industrial customers have used these boards in everything from to self-pour beer taps (**magpi.cc/ipourit**) and coffee machines to electric vehicle charging stations. We've also seen specialised medical monitoring devices (**magpi.cc/biobusiness**) built around our Compute Modules. A market research customer of ours used the modules to develop a system that understands the types of TV programmes different people enjoy watching, and Kunbus has developed an entire industrial product line around our Compute Module, the Revolution Pi (**magpi.cc/revolutionpi**).

You can browse our customer success stories (**magpi.cc/success**) for plenty more examples of all types of Raspberry Pi product in use in business and industry. You'll find everything from farming and factory automation to digital signage and earthquake monitoring. And to make sure you don't miss any news about Raspberry Pi for industry, sign up to receive our quarterly updates for business customers (**magpi.cc/industryemail**). 🅜

◄ The Kunbus Revolution Pi is an industrial product line built around Compute Module 4

sprinklR
irrigation

Keen computer programmer Mark made use of Raspberry Pi to create his own automatic watering system. By **Rosie Hattersley**

MAKER

Mark Niemann-Ross

Mark has written several books about using technology, particularly the R programming language.

niemannross.com

⚠️

Warning!
Water & Electricity

Running water and electrical currents can seriously damage your beloved tech hardware! Follow Mark's lead and encase your plant irrigation device in a waterproof housing such as a sealed glass jar.

magpi.cc/ electricalsafety

Automatic watering systems that tend to your plants' irrigation needs cost only around £30 and work perfectly well. As maker Mark Niemann-Ross points out, they reliably turn on and off, sprinkling thirsty crops when the season demands.

Unfortunately, they score far less highly for longevity: replacing a worn-out system every year or two soon becomes an uneconomical chore. Such a system can be made smarter with internet connectivity, but this adds roughly $100 to the cost, and the system invariably still falls foul of harsh winters. Mark's DIY irrigation project uses Raspberry Pi Zero WH and a cheap, but clever, waterproof housing to ensure it lasts year after year.

Harsh environment

Portland, Oregon-based Mark has an urban garden that needs to be irrigated each summer. Judging from the description of his irrigation system, the garden is a little larger than the 30-foot back lawns often attached to suburban UK homes. Commercially available timers provide reliable irrigation, but Mark found they don't last: "If you forget to bring them in during the winter, they freeze and break. After a few years, the plastic valves wear out and they jam closed (bad for plants) or jam open (bad for water bills). They don't adjust to rain or hot weather." Pricier, more robust timers plus web connectivity to check the weather forecast work better but still fail. Mark had several Raspberry Pi boards at home, and decided to put them to good use. Having written extensively about Raspberry Pi as well as coding

▼ Components and code were tested over several days in a prototype design

Mark's R-based automated irrigation system uses standard irrigation valves from a hardware store

Raspberry Pi Zero WH, kept secure and dry in a sealed upturned Mason jar, checks Open-Meteo's weather forecast and adjusts the water flow accordingly

A single line of R code initiates plant watering once each evening and monitors usage for billing purposes

Quick **FACTS**

> The system runs off a 1 amp 5v phone charger

> It is set to water Mark's plants daily at 6pm

> But checks the weather forecast first

> Mark is now planning to automate feeding his chickens

> With each chicken having its own RFID tag

Irrigation Status

Range of Days

Documentation located at
https://niemannross.com/link/irrigation

MM of Water   Seconds   Status

Sprinkler System Status
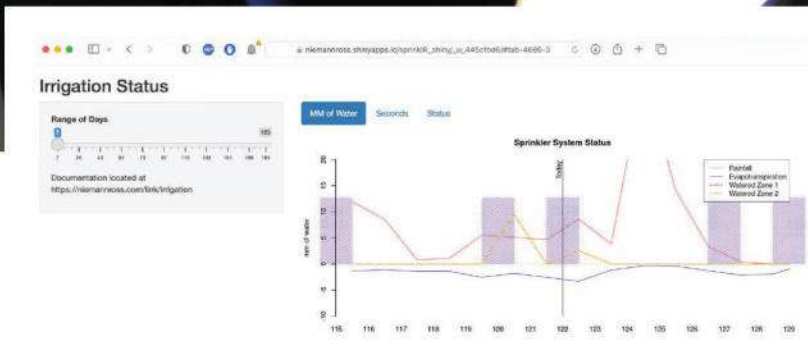
◄ A web dashboard Mark created shows rainfall trends and his irrigator's status

> ❝ Households can reduce their water usage by roughly 7600 gallons a year ❞

▶ Mark designed a robust replacement for a commercial irrigation system

using R (see his blog at **niemannross.com**), the combination seemed obvious, although R is a less common choice of programming language. "My preferred language is R, which I can run from a Linux operating system, but not from MicroPython or C. Irrigation only happens once a day, so I don't need speed. What I need is the most convenient way to express my logic to Raspberry Pi." Using Raspberry Pi Zero WH as the controller "makes it easy to connect to the internet and the headers provide a convenient way to connect relays and buttons," he explains.

Bits and pieces

Creating his irrigation system involved assembling lots of fairly standard components and a certain amount of planning. Prior knowledge of plumbing and electronics is helpful, Mark observes. The planning aspect required Mark to work out how much rainfall was likely to offset the total number of gallons of water his garden would need and how long the valves would need to be open at a time. He calculated the rate at which his house pipes could pump water to the irrigation valves (and the putative number of gallons per hour) as well as the voltage required for the relays to deliver it.

To see whether his idea would work Mark began by screwing irrigation valves, two servos and Raspberry Pi needed to power the system, as well as a Raspberry Pi Zero, on to a piece of wood. "There isn't any water connected to the system at this point – I'm only trying to test the electronics and develop and test the code."

After writing and testing the code on Raspberry Pi he "ran it with the relays for days without any valves connected, then tested the plumbing in a sink before I attached it to the board".

The US Environmental Protection Agency estimates 28 million US homes have an automated watering system, and that households can reduce their water usage by roughly 7,600 gallons a year using a weather-based system to gauge requirements (**magpi.cc/irrigationepa**). Mark wanted to see whether he got realistic readings, and how well the system performed, before deciding to risk linking it up with his water meter for billing. He continues to tweak and update it and is delighted with just how well his $75 irrigation system is performing. 🅜

▲ Daily irrigation helps keep plants in Mark's garden healthy

Thirsty work
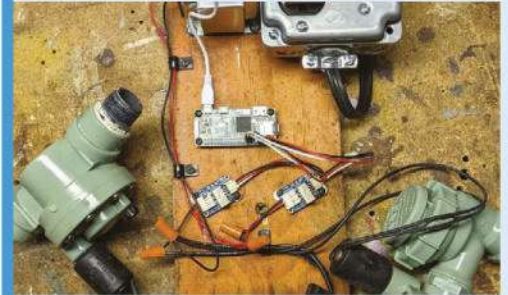
01 Mark documents setting up his irrigation system at **magpi.cc/sprinklrwiki** including setting up the power supply, servos and valves and using R code on Raspberry Pi to implement daily watering.

02 An upturned Mason jar provides a useful means of waterproofing the electronics, with cables fed up through the board and the jar's sealed lid.

Free Weather API

Accurate Weather Forecasts for Any Location

03 Code running on Raspberry Pi pulls weather data from Open-Meteo to help determine how much water the plants need.

Audiophile Pi

Alan Boris' low-cost, high-quality DIY audio streamer hits all the right notes, as **David Crookes** discovers

MAKER

Alan Boris

Alan Boris is a technology executive, entrepreneur, software developer and hardware experimenter who loves Raspberry Pi, home automation and software-defined radio.

magpi.cc/ audiophilepi

Audio streamers don't come cheap. You could easily spend between £500 and £1,000 for a decent high-end model. But, as Alan Boris has found, there's a far less expensive alternative that won't look out of place in a standard audio system. By snapping up an old FM radio stereo tuner – a Sony ST-JX411 dating back to 1991 – and making use of a Raspberry Pi 4, he's been able to create his own, modern network music streamer that can be controlled by buttons on the front panel.

As someone interested in giving old tech a new lease of life, the idea came naturally. "I was inspired by the many high quality network audio streaming devices sold by the major stereo component brands such as Sony, Cambridge Audio, NAD and so on but I couldn't justify the cost," he explains.

> " The Sony tuner has a retro vibe but doesn't look too dated "

▲ Alan would like to add text scrolling to the display for longer titles and to keep it in the selected mode when the song changes

"I figured they just play back digital audio so I started looking at DIY options. I knew I wanted a standard-sized stereo component and decided an old stereo tuner would be reasonably-priced because they are not in very high demand, even among audiophile collectors."

Treble buy

Alan purchased three Sony ST-JX411s so that he could mix-and-match the best looking parts ("most were dented and dirty," he says). He liked its look and noted it had space for a decent-sized LCD. "The Sony tuner has a retro vibe but doesn't look too dated, and it fits in with much of my second hand stereo equipment," he says. "I also thought that a repurposed tuning knob would be a nice touch for a streaming player – few attractive modern tuners have one."

The front panels were important. "I wanted to play a playlist or start and stop the device without using a web interface on my computer or phone," he explains. "Another requirement was for an alphanumeric display for the currently playing song, title and file format. Since my amplifier already had a digital-to-analogue converter [DAC], I wanted quality digital audio – S/PDIF – too."

For the audio quality, Alan chose a HiFiBerry Digi2 Pro which uses the I2S sound port for S/PDIF audio output (without a DAC, he'd have opted for a DAC HAT such as the Raspberry Pi DAC Pro). To connect everything together, he used a small proto PCB board with some male headers. "That way I could use female-to-female jumpers to connect to the Pi GPIO pins," he says.
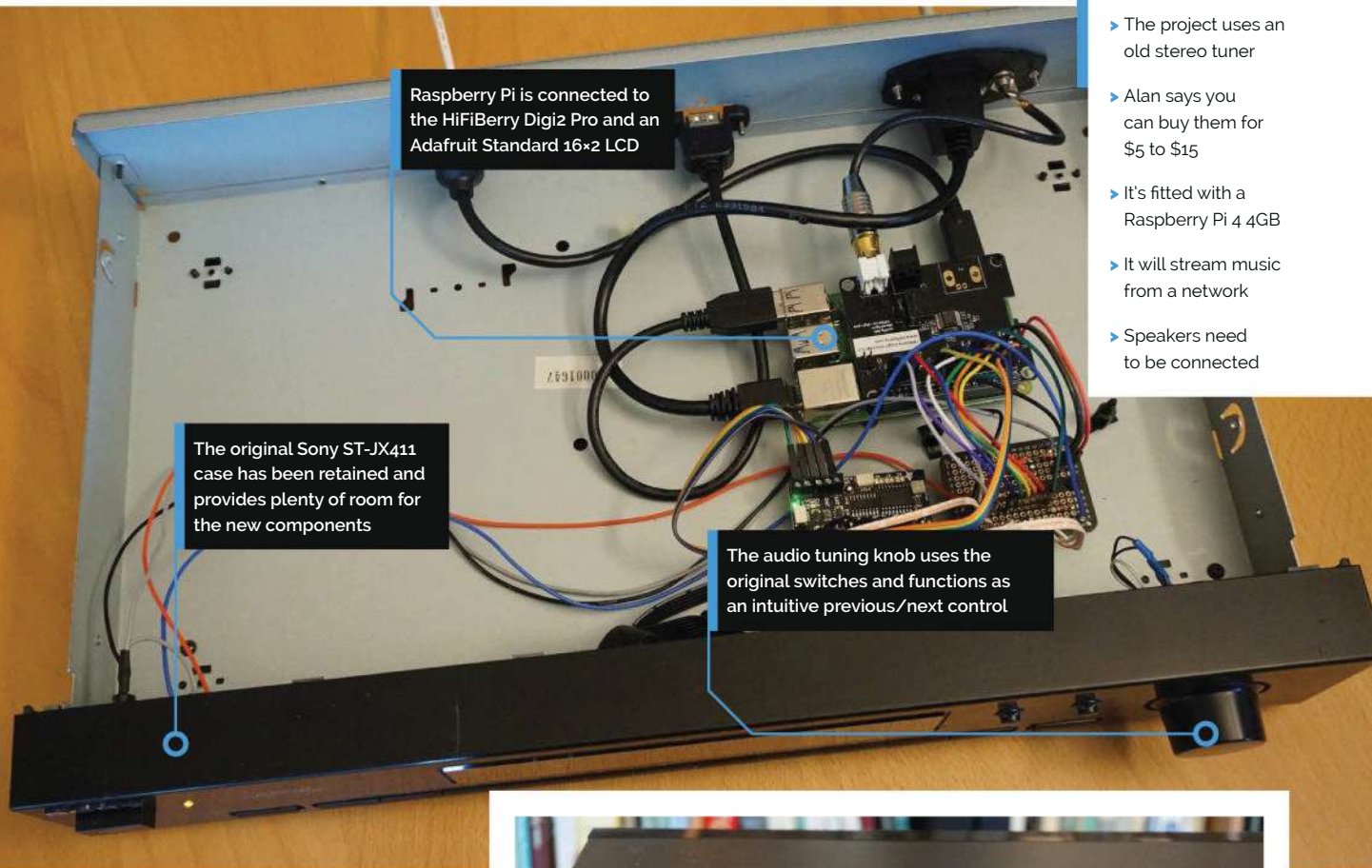
Setting the tone

Alan retained the logic board which houses the front-panel buttons. "The main challenges related to reverse-engineering these buttons and I'm

Raspberry Pi is connected to the HiFiBerry Digi2 Pro and an Adafruit Standard 16×2 LCD

The original Sony ST-JX411 case has been retained and provides plenty of room for the new components

The audio tuning knob uses the original switches and functions as an intuitive previous/next control

not sure I would have accomplished the project without a schematic diagram I found online," Alan continues. As it was, Alan was able to read row and column values, reducing the number of wires from 17 to 11. He also replaced the logic board's vacuum fluorescent with a new alphanumeric LCD, and Raspberry Pi's ports were extended to the tuner's back panel.

To control the functions, Alan used the open-source project moOde. "I had already been using moOde on Raspberry Pi 4 and I was happy with its features and audio quality," he says. "I also liked the simplicity of the moOde API, so I didn't have to spend much time figuring that out."

So how does it fare? Great, as it happens. "It works very well, with no issues," Alan says. "And since I'm using the original board passively – that is with no power applied – there's not as much to wear out or go wrong. I now use the music streaming player almost every day and the audio quality is indistinguishable from the original source material, at least to my ears." **M**

▲ The back of the unit remains as neat as it did when it was made thanks to the use of Adafruit panel mount extensions

◀ Alan, who has connected his streamer to a Loxjie A30 amplifier, is considering re-labelling the front buttons

Mini Dexed

A microcontroller synthesiser emulator running on Raspberry Pi involved plenty of collaboration, discovers **Rosie Hattersley**

MAKER

Probono (aka Simon Peter)

Probono works in product management and is based in Germany, where he's been tinkering with Raspberry Pi projects since their 2012 launch.

magpi.cc/ minidexedgit

Dexed is a multi-format plug-in synth "closely modelled" on the Yamaha DX7 intended as a companion and tribute to the 1983 machine that inspired it. So reads the GitHub entry (**magpi.cc/dexedgit**), which provides multiple useful links indicating just how many directions the concept of this free multi-platform music maker has been extended by audio and electronics enthusiasts.

Mini Dexed ports the concepts of the digital synthesiser to Python and Raspberry Pi, and is the brainchild of Simon Peter (aka Probono, **magpi. cc/minidexedgit**), who describes it as a 'Dexed FM synthesiser similar to 8x DX7 (TX816/TX802) running on a bare-metal Raspberry Pi (without a Linux kernel or operating system)' to produce eight tones. Voices can be programmed using a DX series editor using MIDI sysex.

Emphasising the importance of Yamaha's DX7, composer and MusicRadar journalist OSC Steve argued: "it's possible to categorise 1980s music into two eras; the pre-DX7 era and the post-DX7 era, such was the widespread use of this new instrument and its distinct timbral character".

Bringing DX7 to life

Probono has been experimenting with Raspberry Pi, for home automation and 3D printing as well as digital music, since its launch in 2012. He began the Mini Dexed project because he was keen to make a "real musical instrument" rather than something that felt like a computer. He is in awe of the original developers of FM synthesisers and says Mini Dexed "stands on the shoulders of giants". To recreate their sounds he was looking for a commonly available, inexpensive

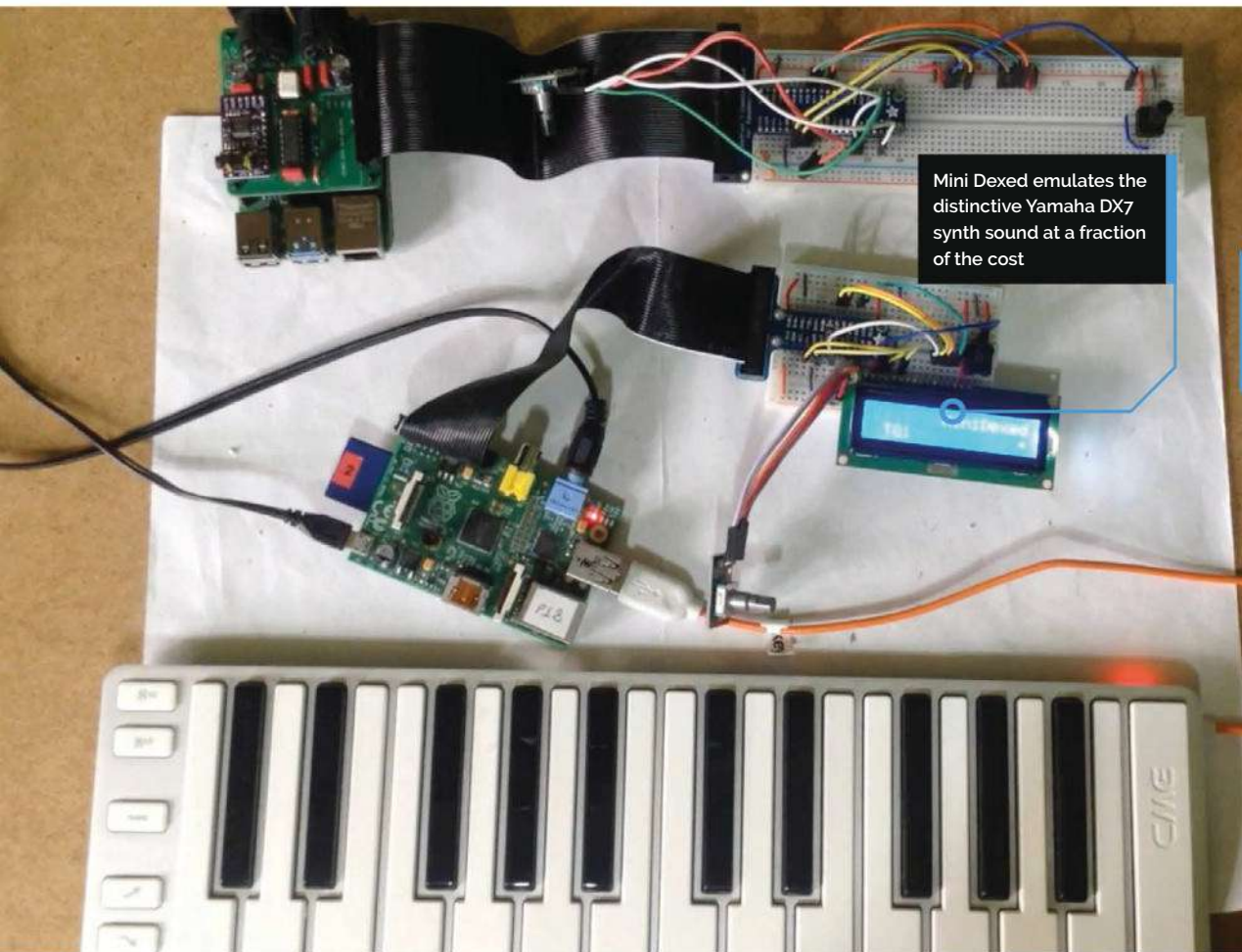▶ Different instrument voices can be called up

microcontroller with lots of computing power and is "using Raspberry Pi more like an embedded microcontroller than a regular computer". Quick boot times, small code size and overall simplicity are further hallmarks. The project began in 2022 during a discussion in the Circle project, when Probono asked for guidance on how to go about integrating an existing synthesiser engine. Another maker, Rene Stange (**magpi.cc/circlegit**) produced Circle, a library for code that runs it in a bare-metal Raspberry Pi environment, while Holger Wirtz (**magpi.cc/picodexedgit)** ported the Dexed synth engine for use with microcontrollers, creating a framework specifically for this scenario.

Future sounds

Mini Dexed is a flexible platform for experimenting with electronic sound. Once it had been ported for microcontroller use, Probono deliberately built Mini Dexed around Raspberry Pi and commonly available, inexpensive hardware components. He says a Raspberry Pi Zero 2 version could be created for less than €50. Hardware choices and whether to use a dedicated audio DAC are down to individual

Mini Dexed emulates the distinctive Yamaha DX7 synth sound at a fraction of the cost

Quick **FACTS**

> There's a useful setup video at **magpi.cc/ minidexedyt**

> The free Mini Dexed is comparable to a 1980s TX816 synthesiser (**magpi. cc/tx816**)

> One cost roughly $5,000 then, or $15,000 now with inflation!

> Maker Holger Wirtz has even enabled a USB version

> Community discussions are at **magpi.cc/ minidexed- discussions**

3V3 MIDI
TX RX

2.2kΩ
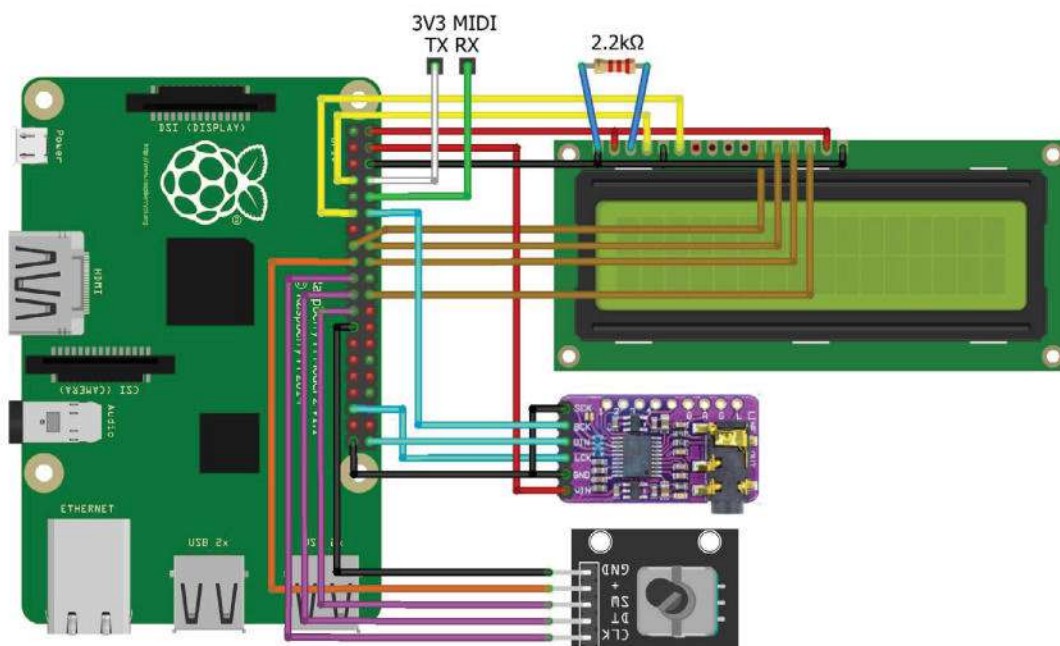
◀ The hardware setup is straightforward

Warning!
Use test hardware

The makers strongly recommend using old or second-hand equipment for your experiments, rather than destroying wonderful old electronic instruments.

magpi.cc/minidexeddx7

> " I could imagine hooking up all sorts of sensors via MIDI to Mini Dexed, in order to create an immersive soundscape that changes as its surroundings change "

makers' preferences, while suitable connections for the audio partly depend on which Raspberry Pi you're using (**magpi.cc/minidexedhardware**). Those that can be configured to use USB Gadget Mode instead of USB Host mode (currently Raspberry Pi 3 and 4 but not yet 5) allow MiniDexed to be used as a USB MIDI device and accept audio streaming from MIDI keyboards, for example. "As someone who is interested in experimental music, I could imagine hooking up all sorts of sensors via MIDI to Mini Dexed, in order to create an immersive soundscape that changes as its surroundings change."
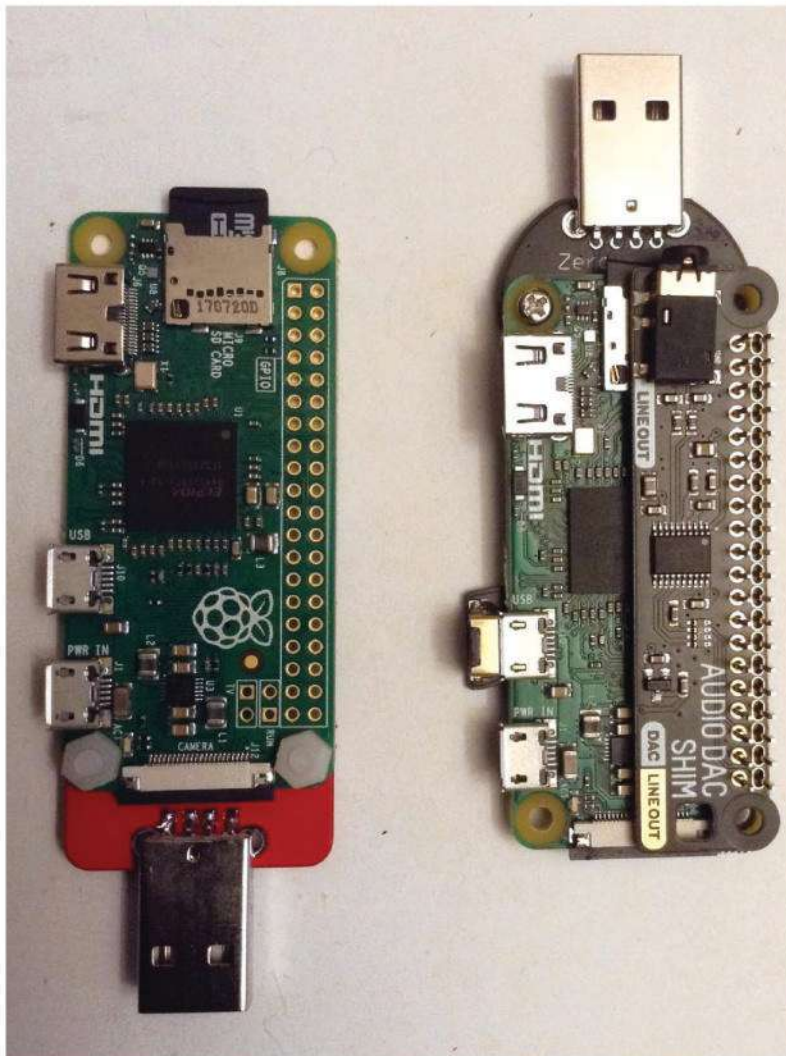
Mini Dexed has also been extended to work with external DACs (making it usable with Pico and Raspberry Pi 1 and 2, too) as well as adapted to support 16 voices by blogger Kevin, (**diyelectromusic.com**) who has contributed to this fantastic synth's development and raised its profile. Probono specifically mentions the potential of Raspberry Pi Pico which Kevin was able to implement: see **magpi.cc/picodexedgit** or **magpi.cc/dx7usbdongle**.

"While the project can be built without any extra hardware, a simple display and a rotary encoder and/or some buttons make it much easier to use, and an inexpensive digital to analogue converter increases sound quality significantly." Probono says "The real cost is the time invested into developing, building, refining, testing, discussing – and the MiniDexed community collectively has put in, and is still putting in, a lot of time and effort, which I am very grateful for."

"I am still trying to wrap my head around how to design sound from scratch using FM, it's probably a learning journey for a lifetime." **M**
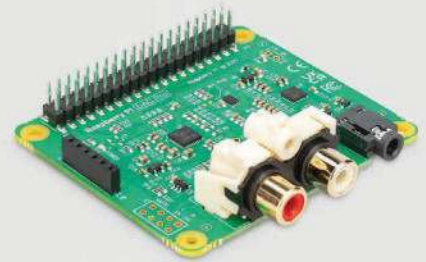

▶ Mini Dexed aims to replicate the iconic 1980s synthesiser sound

Sounds familiar

01 Raspberry Pi 4, 5 or Pico can all run Mini Dexed without booting into the operating system, ideally with a separate DAC such as the one at **magpi.cc/dacpro** connecting via GPIO ports.
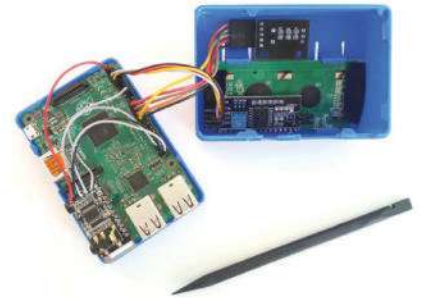
02 A 16×2 LCD display, breadboard and a rotary encoder or joystick are needed to navigate the menu. A microSD card then needs to be loaded with the Mini Dexed code and synthesiser software from Probono's GitHub (**magpi.cc/minidexedgit**).

03 Extract the zip file to the microSD card and save it to Sys Ex, then create a sub-folder named Voice. Connect a MIDI keyboard or USB controller, and power on your Mini Dexed.

▼ DIYElectroMusic created a USB version of the DX7

Cat TV

Prolific maker Becky Stern chose a Raspberry Pi display to entertain her favourite feline. **Rosie Hattersley** wholeheartedly approves

MAKER

Becky Stern

Becky is lifelong maker with interests in art and electronics who loves sharing her creations online.

beckystern.com

Becky Stern is a content creator extraordinaire! During her six years as director of wearables at Adafruit she created, wrote, recorded, presented, and edited 140 videos showing other makers the possibilities of body-worn tech.

Her enthusiasm for tech shines through in her current role at DigiKey, while her personal blog and extremely popular YouTube channel present such gems as Cat TV (**magpi.cc/cattvyt**), a self-explanatory project based on Raspberry Pi Zero and a Elecrow touchscreen. With such a busy companion, we're relieved to hear that Becky's feline friend is being entertained.

Dress to impress

Becky's interest in electronics began at art school. "I have no formal engineering background, but I did drop out of a PhD programme," she explains. Becky was already working at Adafruit when Raspberry Pi came out in 2012. The company immediately saw its potential for makers and for wearable tech: "I remember we delayed the FLORA launch to focus on Raspberry Pi content," says Becky. Her numerous makes include Raspberry Pi ones, of course, including an automated GIF camera (**magpi.cc/gifmaker**) and a smart mirror: **magpi.cc/beckymirror**.

While chasing his own reflection in the mirror might have diverted her cat, Benchley, for a while, Becky thought it was about time he had something tailored specifically to his interests. "My cat loves watching TV. The Raspberry Pi streams his favourite YouTube channel, full of birds and squirrels." Other feline goggleboxers may already be aware of Birder King (**magpi.cc/birderking**), which is also beloved of parrots and dogs.

▼ Benchley sometimes tries to locate the wildlife show on Cat TV

▲ Raspberry Pi Zero and an Elecrow touchscreen form the project's basis, all wrapped in a custom enclosure

Tiny telly

Becky chose Raspberry Pi 5 as the basis of Cat TV, since she wanted to stream video and wanted the extra processing power, but she has also tested it with Raspberry Pi 4. She had a five-inch touchscreen she was keen to use: "I knew I wanted to use this particular display from Elecrow, so I 3D-modelled an enclosure that would fit to its dimensions." Becky used Tinkercad to design a CRT-style cabinet complete

> **"** My cat loves watching TV. The Raspberry Pi streams his favourite YouTube channel, full of birds and squirrels **"**

with ventilation holes so the miniature TV setup looked authentic (**magpi.cc/tinkercadcat**). "I used the honeycomb shape generator in Tinkercad to make grids of holes that are too small for my cat to reach through," she says. A bit of glitz was added by using Galaxy Black PLA filament for the front of the 3D enclosure. The whole project cost roughly $100, and Becky was able to get all the electronics she needed from DigiKey (where she works) including a USB speaker and keyboard. Due to Cat TV's miniature proportions, she had

Becky's cat Benchley relaxes by watching lots of TV

The Elecrow touchscreen hides a Raspberry Pi 5 which streams Birder King wildlife footage all day long

Benchley sometimes peeks behind the 3D-printed TV cabinet looking for the mice and birds he's seen on TV

Quick **FACTS**

> Benchley is named after the writer of Jaws

> Given his enviably stress-free existence

> ...we've renamed him Riley

> Becky made her cats a Raspberry Pi-powered automated food bowl too

> She believes the 1988 film Scrooged predicted Cat TV

to use a right-angle USB connector and a low-profile HDMI to micro HDMI adapter.

Once she was happy with how the TV cabinet and screen fitted together, she installed and set up the electronics, taping the display in place in case she needed any adjustments based on feedback from the ever-watchful Benchley. Becky explains that Cat TV doesn't use any software besides Raspberry Pi OS: "I boot up, open the web browser, navigate to YouTube, make the video fullscreen, then unplug the keyboard. If I were to add some software, I'd write a startup script that performs the same steps."

During trials, Becky discovered the touchscreen was a bit of a problem as the cats would paw the screen and pause the video, so she replaced the screen's USB cable with a charge-only lead, disabling the data connection for the touchscreen. Once this was done she sealed the case and presented it to Benchley and his brother Hamlet. "Cat TV gets used every day! The cat loves it, though it would be cool to add some way for him to change the channel." M
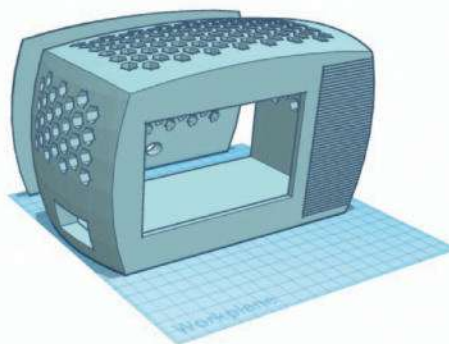
▲ The Birder King YouTube channel streams cute wildlife all day long

◀ You can download Becky's 3D TV cabinet design from Tinkercad

RC plane OSD

Upgrading your RC plane is easy when you can get a Raspberry Pi Pico to give you a HUD. **Rob Zwetsloot** gives it a go

MAKER

Wojciech Domski

Wojciech has a PhD in robotics, and builds embedded systems for professional applications for fun. He's been interested in RC planes since 2000

magpi.cc/rcosd

▼ The resulting image from a flight with the OSD

Remote and wireless technology has come a long way in the last decade or so. With the explosion of drones with cameras, and the continual shrinking of camera tech thanks largely to mobile phones, RC planes have been able to take advantage of the trend too. Maker Wojciech Domski wanted to take it a little further, and decided to overlay data on the picture being sent back.

"It is all about telemetry and tracking what is happening to the plane during flight," Wojciech told us. "For example, the current battery voltage. It is a pretty good indicator of how much power you have left and, thanks to it, you can estimate the remaining flight time."

Pico Power

While looking for OSD (on screen display) devices that would fit his need for telemetry, he came across a project on an RC forum that worked.

"The idea was compelling because it did not use many external elements; only a couple of resistors, capacitors, and diodes," Wojciech explained. "However, it wasn't going to work for me. I
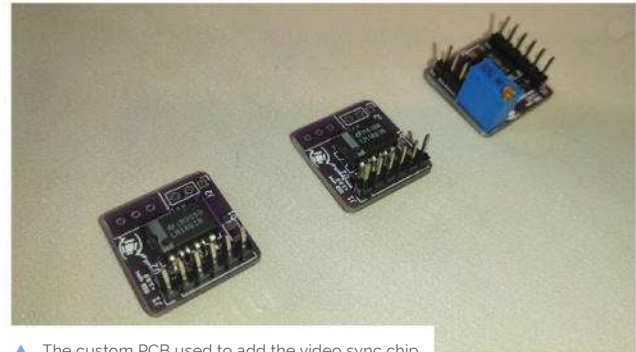
▲ The custom PCB used to add the video sync chip

have carried out tests on different cameras with different lighting conditions and it turned out that without adaptive detection of the synchronisation signal from video, this solution would not work in my case."

Making use of an external video sync signal detector, Wojciech was able to sync up everything needed and just needed a way to get the flight controller to talk with an OSD. The I2C bus ended up being the solution.

However, instead of using Raspberry Pi for the OSD, Wojciech decided instead to experiment with Pico: "I thought that an OSD project using Raspberry Pi Pico would be a good starting point."

"I decided to use it because of the nice features it has," Wojciech continues. "For starters, the amount of flash memory compared to other microcontrollers but also the PIO. I think it is a very nice and useful feature that has been lacking in other microcontrollers."

He also mentions that its size helped a lot as well. Along with a custom PCB with the video sync chip connected, he created the OSD.

Flying higher

"Since using the build, I have not had a single problem with the OSD device," Wojciech tells us. "It is reliable and works under different lighting

Video is played back on a separate screen for the user so they can see where they're flying, along with flight data

Data and other telemetry are sent to the controller

A camera attached to the nose of the aircraft allows for a first-person view

Warning!
Drone Safety

Be careful when working with drones, especially those with spinning blades. Fly safely and responsibly and follow local laws and regulations.

magpi.cc/dronesafety

Quick **FACTS**

> The video sync chip used was an LM1881

> Wojciech has been using Pico and RP2040 for about two years

> Wojciech uses Raspberry Pi elsewhere in a university setting…

> … which allows students to program microcontrollers remotely

> The video system is analogue, which adds extra complexity

> I thought that an OSD project using Raspberry Pi Pico would be a good starting point

conditions. Also, I believe that the most important part of each new feature is the ability to turn it off. When flying, I sometimes disable the OSD in order to have a full view of the video, without any additional information like text."

While Wojciech seems satisfied with the result, there's other ideas he has for the OSD – he reckons Pico has enough power left to display more text and even draw shapes and graphics live.

"Another feature would be add a dimming option to the text," Wojiciech says. "It would add some shadow behind the printed text. This would significantly increase readability. However, this would require some more modifications not only to the code but to the hardware as well." M

▲ With analogue video the quality can deteriorate

EpiSensor energy management

Securing energy services' IoT infrastructure with Raspberry Pi for a more sustainable future

Over the coming years, millions of energy-consuming and producing things will be connected to the internet to enable a new energy economy and facilitate a rapid transition to sustainable energy. Traditional metering, control, and automation systems were never designed to address this problem – they require a high level of technical expertise to deploy, and are too complex and expensive – slowing down energy services companies.

EpiSensor was founded in 2007 to make it easier for energy services companies to scale quickly and communicate with remote sites and assets, bringing a 'consumer class' user experience to a high-quality, industrial product range that can meet the highest standards of accuracy, security, ruggedness, scalability, reliability and interoperability.

Aiming to accelerate the transition to clean, sustainable energy, the company provides an internet of things (IoT) infrastructure that allows its partners and end customers to easily monitor energy consumption and control remote sites and assets, so grid operators can integrate more renewables, and people can make more informed decisions that improve efficiency, reduce costs, and reduce environmental impact.

With a world increasingly switched on to the need to be more energy-responsible, EpiSensor has the power to transform electricity usage and spend in a variety of commercial and industrial environments.

The challenge

Global energy consumption grows year on year, but it's estimated that around 70% of all energy produced is wasted due to inefficiencies. Figures also suggest that about three quarters of our energy comes from sources that emit greenhouse gases. It's why there is a global ambition to reach net zero carbon emissions by 2050, and why accessing more renewable energy is important.

"Over the coming years, millions of sites and assets will need to be connected to the Internet as we rapidly transition to sustainable energy. We have the technology today to run our electricity grids on 100% clean renewable energy, but it needs a dedicated and specialised communications infrastructure, so energy-consuming and – producing 'things' have awareness of the conditions on the grid in real time – because traditional systems were never designed for that job," says Brendan Carroll, CEO of EpiSensor.

EpiSensor's team had examined traditional providers and noted that despite being accurate, reliable, and secure, those systems tended to be closed, complex, difficult to maintain, and expensive. Building that specialised infrastructure has meant starting from the ground up to integrate and automate every component, and remove points of friction that were slowing down energy services providers – without sacrificing the qualities associated with traditional systems.

Crucially, they wanted it to be easily scalable so that organisations regardless of size would be

▼ EpiSensor's Gateway routes data from networks directly to the IoT platforms of its customers and partners

able to create a system that addressed their needs. They also wanted it to be easy to use, bringing a consumer-class experience to the systems so that they would be widely accessible and easy to adopt.

More recently, the company has focused on delivering high-end demand response or demand-side flexibility, which can assist electricity grid operators with balancing supply and demand by incentivising customers to reduce their consumption during peak periods. It is a critical tool in energy transition, reducing strain on the grid, and EpiSensor sought to be at the forefront by designing and building highly accurate, scalable, and secure IoT infrastructure for demand response.

While considering all of this, EpiSensor also knew that data needed to be handled securely. "Data security in commercial IoT programmes is critical to protect sensitive information, prevent unauthorised access, mitigate cyberattacks, preserve user privacy, comply with regulations, maintain business reputation, ensure operational continuity, and protect intellectual property," says Carroll. EpiSensor discovered that Raspberry Pi would allow them to achieve all of their aims.
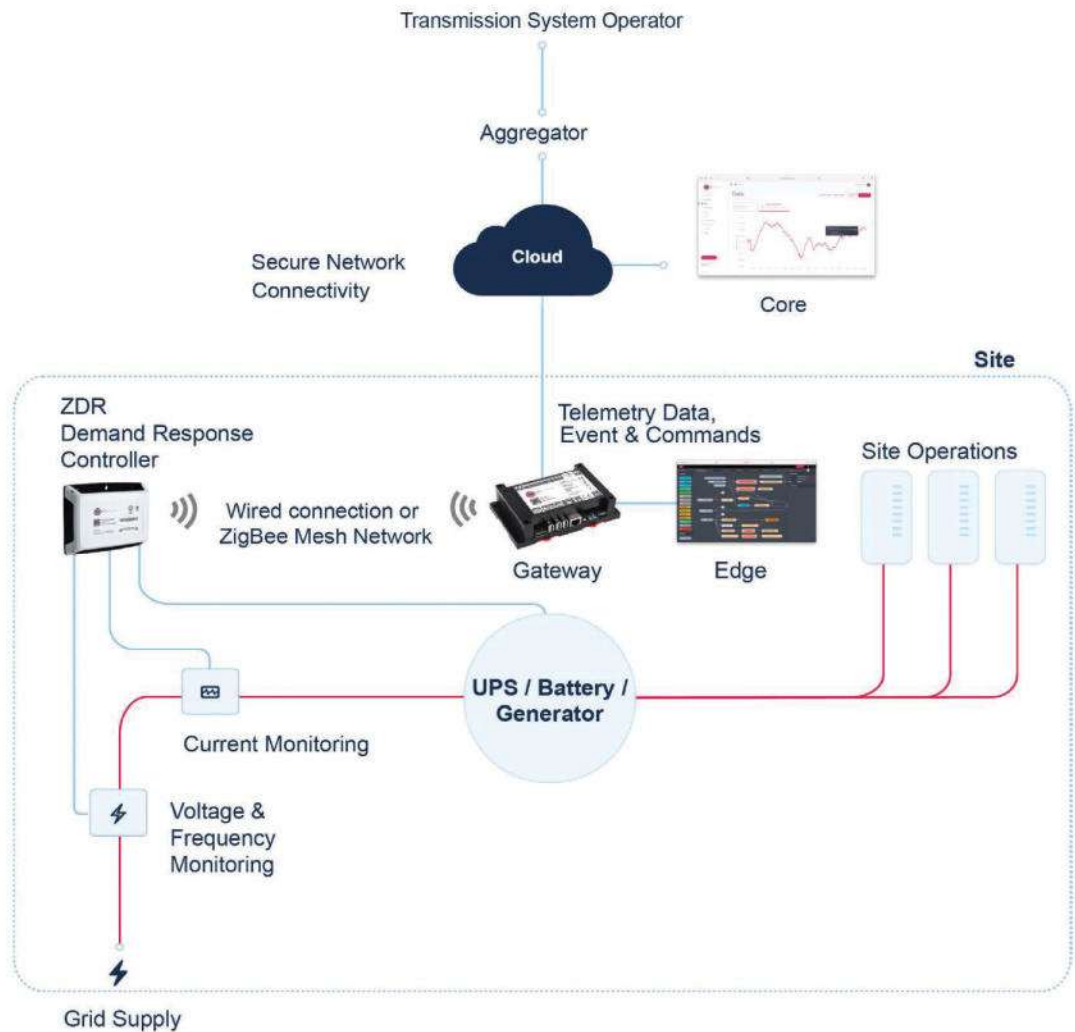
The solution

When EpiSensor was founded in 2007, it was an exciting time for the technology sector. "We saw three new technology launches that felt important: AWS marked the start of a new class of cloud computing, iPhone represented a revolution in mobile and user experience, and new standards like ZigBee made it possible for wireless sensors to communicate securely and at very low power," says Carroll. "We set out to apply this new technology stack, that later became known as the internet of things, to the world's energy and efficiency problems." Over many iterations based on customer requirements, this led to the creation of EpiSensor's latest API-enabled Gateway built around Raspberry Pi Compute Module 4.

EpiSensor's Gateway is a compact device that routes data from networks of wireless sensors directly to the IoT platforms of customers and partners. "It's a powerful, rugged, and secure embedded computer with an IoT app store and many wired and wireless communications interfaces: ZigBee, Wi-Fi, Bluetooth, 4G cellular, LoRa, GPS, CAN bus, RS-485," Carroll says. EpiSensor has worked hard to automate many of the configuration steps that would normally be done by experts on customer sites, so the Gateway essentially configures and tests itself. "EpiSensor's

▲ EpiSensor turned to Raspberry Pi Compute Module 4 to create the IoT infrastructure layer for energy services, helping to accelerate the sustainable energy transition across the world

Gateway is at the core of our advanced IoT solutions," Carroll says. "It's secure, developer-friendly and has data available in open formats."

The huge benefit of EpiSensor's IoT technology is that it can easily be deployed and integrated with systems on customer sites such as existing building management systems (BMS) or SCADA, as well as systems running in the cloud that provide complete energy management or demand response solutions. As well as a set of tools that make building custom integrations quick and easy, partners can leverage EpiSensor's comprehensive list of pre-built integrations to get connected to all of the leading energy services platforms with a few clicks.

As the Gateway gathers data from wireless nodes in real time, the information is made available via the Gateway web interface and mobile apps. It allows all of the nodes and sensors to be remotely monitored and managed via the API of the Gateway, or using Core, EpiSensor's device management platform, making it easier for them to run multiple energy services programmes for their customers.

The user is able to view real-time information on the performance of the system, adjust configuration easily, and visualise how all of the nodes are connecting over the wireless mesh network.

"Our objective is to be ten times quicker and easier to deploy than traditional systems which means our partners can scale energy solutions faster," says Carroll. "We enable facilities managers, system integrators, energy service companies, energy managers and software as a service providers to deliver world-class energy management, demand response, and environmental monitoring programmes, with low cost, easy-to-deploy IoT infrastructure."
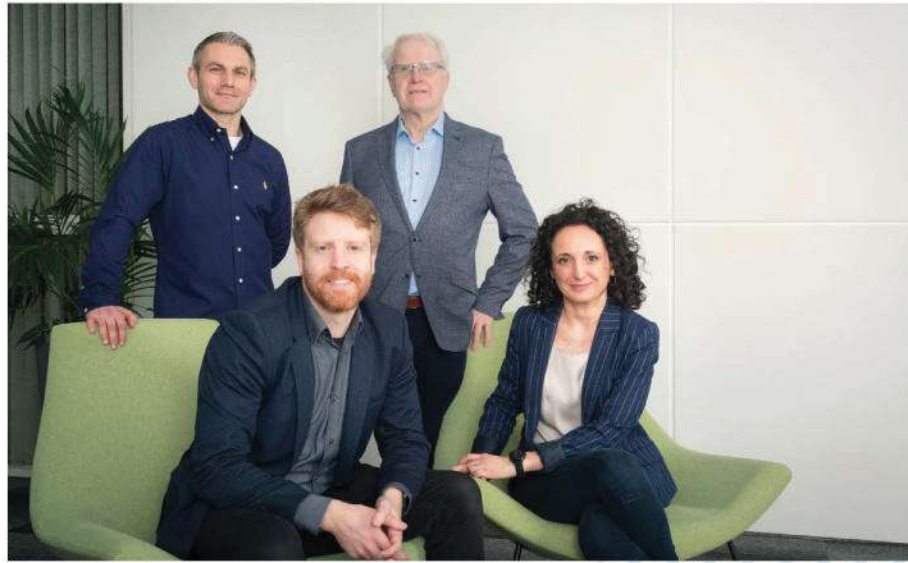
Why Raspberry Pi?

Since EpiSensor was founded, its expertise has grown. "We've attracted a world-class engineering team based in Ireland and we are led by our experienced management team who have grown and sold multiple high-tech companies in related industries in the past. Growing at pace now, our

IoT infrastructure is relied on by market leaders worldwide," Carroll says.

Rather than design the computer module at the centre of their system architecture from scratch, they realised Raspberry Pi offered a simple yet powerful solution. "EpiSensor opts for Raspberry Pi due to its cost-effectiveness, reliability, and superior performance, but most importantly the invaluable support of its thriving community," Carroll says. "If a large community is using a computer board like the CM4 from Raspberry Pi, and you come across an exotic issue that only happens under certain circumstances, there's a higher chance someone else in the community has already solved it, and that makes Raspberry Pi our preferred choice."

Raspberry Pi Compute Module 4 is certainly powerful. It incorporates a quad-core Arm Cortex-A72 processor and it allows Canonical's highly secure and reliable Ubuntu Core operating system to be run; this includes an IoT app store, meaning the functionality of the EpiSensor Gateway can be extended and customised to include additional functionality like machine learning, device management, and advanced

deployments streaming data every second of every day, all over the world," Carroll says.

EpiSensor is well positioned for future growth, with Raspberry Pi enabling the company to nimbly meet the needs of an expanding and demanding market. According to Statista, the Energy Management market worldwide is projected to reach a revenue of $10 billion in 2024, growing at an annual rate of 10.75%. Growth is driven by several factors including rising global energy costs, regulatory change (Energy Efficiency Directive Recast 2023 in the EU, for example), environmental concerns, the race to net zero, renewable energy integration, and energy security.

> **" EpiSensor is well positioned for future growth, with Raspberry Pi enabling the company to nimbly meet the needs of an expanding and demanding market "**

networking applications. It also operates over a wide temperature range, and the company likes that it can operate silently and with low power, while offering passive cooling – which means higher reliability in harsh environments.

The results

EpiSensor maintains business-critical systems with hundreds of energy services companies worldwide across more than 20 countries, and testimonials are glowing. The company's products are praised for their scalability and usability. They are also effective and easy to deploy and commission on site.

"Our partners include Enel X, the world's largest Demand Response Aggregator; Veolia, a world-leader in energy and facilities management; Capula, joint-owned by energy giants Dalkia and EDF and a leader in systems integration; and CoolPlanet, a specialist in commercial decarbonisation. Together with our partners, we have large-scale

Similarly, demand response is growing rapidly driven by challenges faced by grid operators in managing the intermittency caused by renewable energy integration and the surge in energy demand. "The task of providing grid flexibility at the asset level within milliseconds and aggregating multiple assets into Virtual Power Plants (VPPs) through demand response programmes is a significant undertaking, one that we have taken on with the help of Raspberry Pi technology," remarks Cristina Coffey, CSO, EpiSensor. VPPs aggregate various distributed energy resources (DERs) such as batteries, UPS systems, EV chargers, industrial machinery, and even small-scale demand-side resources such as smart appliances or HVAC systems. Their adoption is growing and they are seen as a critical component in the future of sustainable energy. M

▲ EpiSensor's team examined traditional providers and found those systems tended to be closed

SUBSCRIBE TODAY FOR JUST £10

Get 3 issues + FREE Pico W

Subscriber benefits

▶ **FREE Delivery**
Get it fast and for FREE

▶ **Exclusive Offers**
Great gifts, offers, and discounts

▶ **Great Savings**
Save up to 23% compared to stores

Subscribe for £10

▶ Free Pico W
▶ 3 issues of The MagPi
▶ £10 (UK only)

Subscribe for 6 Months

▶ Free Pico W
▶ 6 issues of The MagPi
▶ £30 (UK) $43 (USA)
  €43 (EU) £45 (Rest of World)

📞 Subscribe by phone: **01293 312193**

✈ Subscribe online: **magpi.cc/subscribe**

✉ Email: **magpi@subscriptionhelpline.co.uk**

Subscribe for £10 is a UK-only offer. The subscription will renew at £15 every three months unless cancelled. A free Pico W is included with a six-month subscription in USA, Europe and Rest of World.

SUBSCRIBE TODAY AND GET A
FREE Raspberry Pi Pico W

Subscribe in print today and get a **FREE development board**

- ▶ A brand new RP2040-based Raspberry Pi Pico W development board

- ▶ Learn to code with electronics and build your own projects

- ▶ Make your own home automation projects, handheld consoles, tiny robots, and much, much more

WORTH $6

This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time.

Hidden inside is a Raspberry Pi 5 and 1TB NVMe drive

The Argon ONE has a built-in IR receiver and a dedicated remote

BUILD A RASPBERRY PI 5 MEDIA PLAYER

Free your films, videos and music from your computer with our complete guide

PJ Evans brings the popcorn

Since its inception, Raspberry Pi has always been a favourite choice for those wanting to build their own media centre. But why bother when so many off-the-shelf options are available? There are many reasons: Control over your data, avoiding subscription costs, accessing more obscure services like public domain films and homebrew games. The biggest reason of all is because it's fun. Over the next few pages we'll take a look at the hardware and software available to you, and run through a powerful setup using the popular Kodi platform. Soon you'll be on the sofa watching all your favourites.

HARDWARE

There are a few things you can do to really make your media centre shine

Raspberry Pi 5

Yes, it sounds obvious but the latest iteration of Raspberry Pi is perfect for video performance up to 4K.

magpi.cc/raspberrypi5

£58

Argon IR Remote

Pair your Argon case with this elegant IR controller and you don't have to leave the sofa.

magpi.cc/argonremote

£9

Raspberry Pi TV HAT

Did you know you can watch over-the-air digital broadcasts? Just add this HAT!

magpi.cc/tvhat

£21

Argon ONE V3 M.2 NVME Case

Argon's latest for the Raspberry Pi 5 features M.2 SSD support, active cooling and an IR receiver.

magpi.cc/argononev3m2

£46

52Pi NVDAC

Audio quality your priority? This two-in-one HAT gives you an M.2 SSD and a high-quality DAC.

magpi.cc/52pinvdac

£29

Mini Bluetooth Keyboard

If you fancy browsing the web or chatting from your chair, get a wire-free mini keyboard.

magpi.cc/minikeyb

£13

SOFTWARE CHOICES

Now you have your hardware, what to run on it? Luckily there's no shortage of options

LibreELEC & Kodi
libreelec.tv

- Dedicated image, so easy to install
- Handles videos, audio and more
- Community supported with hundreds of plug-ins

LibreELEC is a dedicated operating system designed to only run Kodi, which is one of the most popular media players available. A solid all-rounder that handles hundreds of different audio and video formats, it's designed for the big screen and works brilliantly with a remote control. Access to streaming comes through the huge repository of plug-ins.

Volumio
volumio.com

- Audiophile-grade solution
- Beautiful interface
- Wide range of remote control apps

If you're looking for a high-quality audio-only solution, Volumio have spent years developing their audiophile solutions for Raspberry Pi. Paired with a good DAC, you have a music player with impressive library management that rivals devices that cost several times more. It's also available as an OS image for simple installation.

Plex
plex.tv

- Perfect for streaming both home and away
- Lots of free content
- Superb library management

Where Kodi is for consuming media on a dedicated device, Plex is a server that gives you access to your media anywhere. Install the server and add your media using the excellent web interface then you can use the Plex client app on your computer, phone or even Kodi. You can also access your media when you're outside your home network.

OSMC
osmc.tv

- Elegant interface
- Highly customisable
- Lots of plug-ins

One of the joys of open-source software is that anyone is free to 'fork' an existing codebase and build their own version. OSMC forked Kodi so they could add new features. The result is an elegant design which suits those of a minimalist persuasion. Like LibreELEC, it's available as an OS image. Choose this if you want a calm, refined experience from your media centre.

FILL UP
THAT SSD

There's no point having a state-of-the-art media player with nothing to play. Here's where you can find great content

YouTube   Videos

The most popular video service in the world is obviously going to be a must-have for your media server. YouTube on Chromium works out of the box if you're using Raspberry Pi Desktop. If you want to use it with Kodi, the good news is a plug-in is available directly from the Kodi interface. Just head over to the Add-ons menu and search for 'YouTube'.

Netflix (also Disney+ etc)   Movies & TV

Many on-line streaming services use Widevine, which is a browser-based anti-piracy system. This is not installed by default on Raspberry Pi, so services such as Netflix won't work. Luckily it can be installed simply, just follow this guide: **magpi.cc/widevine**. If you're using Kodi, try the unofficial plugin: **magpi.cc/netflixkodi**

Qobuz   Music

Over the recent years, music file size has stayed the same, but storage has got bigger and cheaper. There's no longer the need to crush the quality, instead you can use FLAC which is a 'lossless' format that loses no music quality in its compression. The strangely named Qobuz is a leading online retailer of FLAC-encoded music.

DON'T MANGLE YOUR MEDIA

Keep on top of some basic housekeeping, and you'll always be able to find what you want

Tag all the things

The key to a neat media library is to tag everything. Tags are metadata encoded in the media file itself that tells the media library things like titles, artists, genres and track numbers. Without properly tagged files, you may find things hard to find and out of order. Try **magpi.cc/kid3**.

What's in a name?

Quite a lot, as it turns out. Many media libraries will make assumptions based on the filename of your media. For example, Plex likes music files to start with the track number (e.g. "01 Help.flac") and will correctly order them. TV Show filenames should contain the season and episode in the format "s01e02".

Use the scraper

Platforms such as Kodi and Plex have built in 'scrapers' or 'agents' that will look at your media filename and try to tag it for you (although they don't always write the tags to the file, only their database). This can be great for automatically getting artwork and extra fields like actors, directors or lyrics.
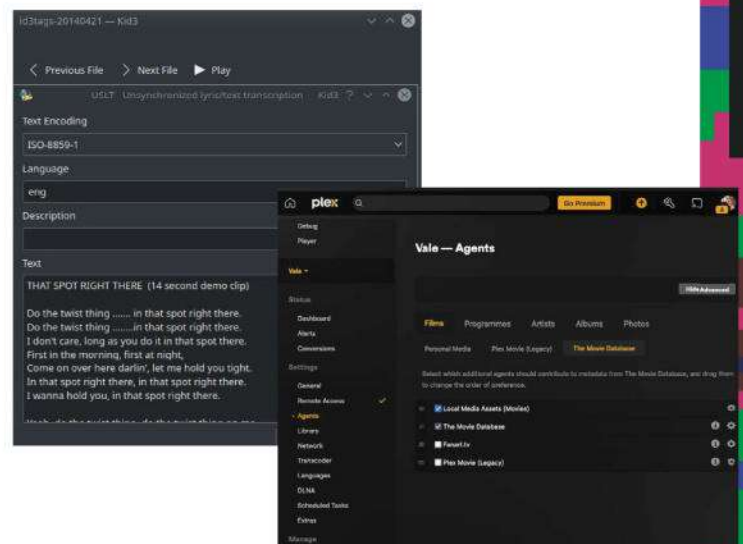
BUILD A KODI
MEDIA CENTRE

If you're not sure where to start, this tutorial will help you build
a powerhouse Kodi media centre perfect for the living room
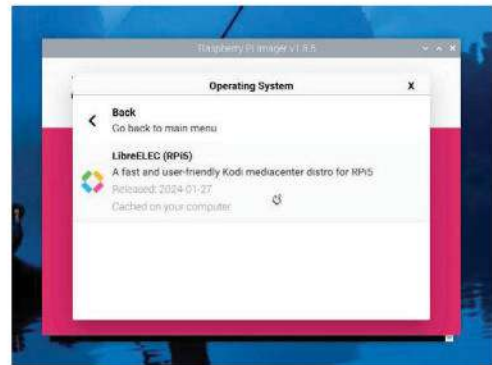
01 Assemble your hardware

Following the provided instructions, mount your Raspberry Pi 5 into the Argon ONE case. Prepare a microSD card with a copy of Raspberry Pi OS with Desktop and insert it into the Raspberry Pi. Carefully connect the NVMe ribbon cable. Connect a monitor, keyboard and mouse then boot up. To enable the NVMe SSD, run `sudo raspi-config`, select Advanced Options, Bootloader Version then Latest. Save and reboot.

You'll Need

> SD Card
> Argon ONE V3 M.2
> rgon remote control
> M.2 NVMe SSD (We used a 1TB)

### 03 Initial configuration and file transfer

On first boot Kodi will start a setup wizard where you can configure things like Wi-Fi and ssh access. Now it's time to load up some local media or configure network connections. For local storage you can map the content folder to a local mapping using Samba or use `scp` to copy files across on the command line (hostname: libreelec.local). Kodi has already created some folders for you in the root directory for movies and music. Start by uploading some tunes into the 'music' directory.

02 Install LibreELEC

Once rebooted, In the Desktop, run the Imager (you'll find it under Accessories) and select 'Raspberry Pi 5' as the device, 'LibreELEC' as the OS (under 'Media Player OS') and select your SSD as the storage. Once the image is written, shut down and remove the microSD card. Put everything back together and boot the Raspberry Pi. Make sure you've got a keyboard connected for the next bit.

" A great money-saving way to watch classic films is to investigate the world of the public domain **"**

Public domain content

A great money-saving way to watch classic films is to investigate the world of the public domain. These are films that have lapsed out of copyright so are free to view and download. You may have to wallow through some stinkers but there are some genuinely brilliant movies out there. Popular repositories are:

- **archive.org**
- **magpi.cc/pubdomfilms**
- **publicdomaincinema.net**
- **magpi.cc/retroflix**

Gaming

Many media streaming services now include gaming. Raspberry Pi is no different. Kodi offers a range of gaming plug-ins including many popular retro computer emulators. Plus, thanks to Microsoft's Xbox Cloud Gaming service, you can play the latest AAA console games in your living room using a Raspberry Pi. For a full tutorial have a look at issue #136 (**magpi.cc/136**).

04 **Configure libraries**

The default folders are already 'mapped' to the different categories in Kodi, but you need to initiate the first scan. Select 'Music' then 'Enter Files Section'. You'll see 'Music' next to a disc drive icon. Right-click on the icon and select 'Scan item to library'. Your music will now be indexed by Kodi and within a few seconds you'll be able to return to the Music library and see all your cover art. Repeat this process for your movies and videos.
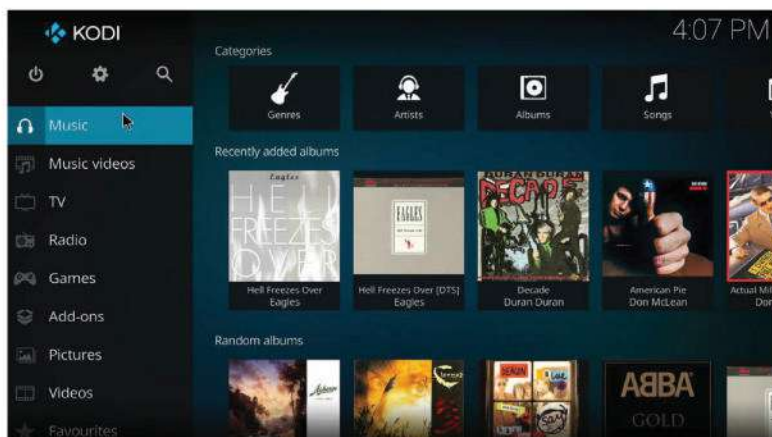
05 **Add remote control**

So you can get all cosy on the sofa and enjoy your media, the Argon ONE case has an infrared receiver. With the Argon remote, you can control Kodi from across the room. Instructions are provided on how to configure the remote here: **magpi.cc/argonremoteconfig**. Once installed, you'll no longer need your keyboard and mouse and your new device will be ready to take pride of place in your media setup. **M**

Part 04:

Learn Python:
discover data types and build a budget tracker

Storing and using data is at the heart of programming. Discover how to use various data types to build interesting programs

MAKER

Lucy Hattersley

Lucy is editor of *The MagPi* and was taught quite firmly to keep an eye on her pennies when young. It's a northern thing..

magpi.cc

Data isn't just the name of an old android in *Star Trek*, it's also the heart of computer programming. The word simply means "an item of information" from its singular Latin "datum" (meaning "that which is given").

Programs largely revolve around data. Obtaining it from input (often from the keyboard and mouse), storing it, manipulating it, and then sending it back in the form of output (typically to the screen).

Right down at the metal, everything is just electronically flipped on and off switches, but that is far too complex for everyday use so everything is abstracted up to a level where we can comfortably deal with it. And part of this is data segmented into different types: some of these will be familiar: strings and integers (numbers). Some are more esoteric: bools, dictionaries, and tuples, for example.

In this tutorial, we cover various data types in Python so you can learn to identify them when you see them. Then we will use these to create a budget tracker program, that uses different data types together to help users manage their money.

We separate data in Python into two different types. The first is Basic and the second is Complex.

Basic data tends to be a single item: a number, word, or statement. Complex data tends to be collections of items, such as lists (that can contain other data).

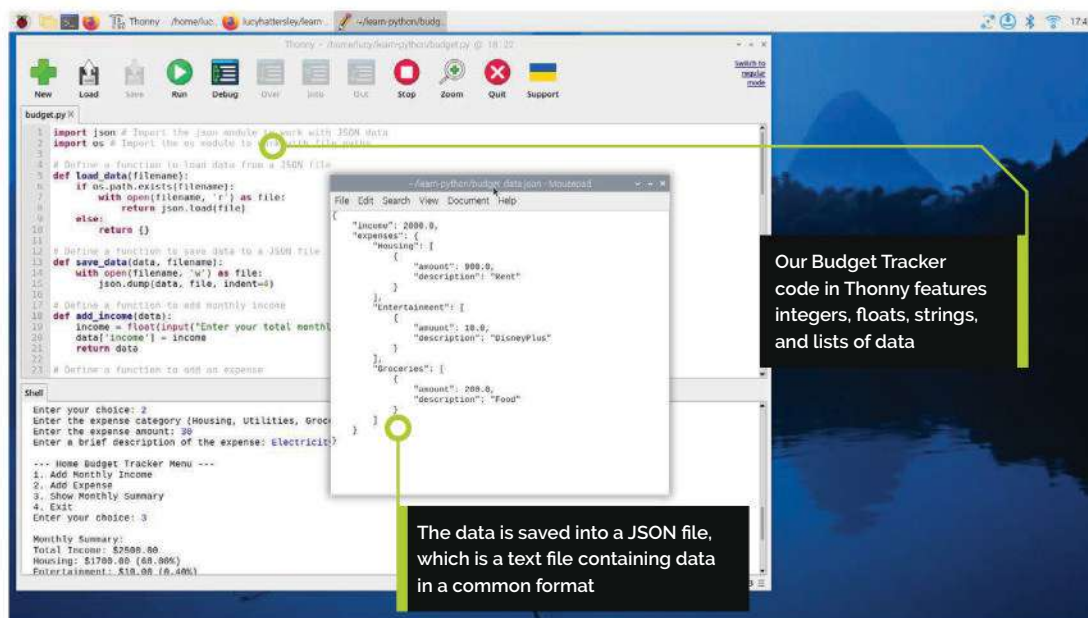Let's start with the most basic of all data types: numbers.

Integers and Floats

There are two types of numbers we're going to look at here. Whole numbers are called "integers" or "ints" and fractional numbers with a decimal point, which are called "floats". This is short for "floating point number".

Open Thonny IDE and enter:

```
$ foo = 42
$ bar = 4.2
```

… in the Shell window below (press **ENTER** after each line). We'll just type our test data into the Shell for now. Check the values with `foo` and `bar` in Shell. It will return "42" and "4.2".

You can check the type of data using the `type()` function. This is handy because – in Python – different types react differently when used. Enter:

Our Budget Tracker code in Thonny features integers, floats, strings, and lists of data

The data is saved into a JSON file, which is a text file containing data in a common format

Top Tip 👍

Why not float everything?

What is the point of ints? Why not just make 42 into 42.0 and have everything as a float? The answer lies deep in the binary hardware. It is easy to represent whole numbers in binary, but much more complex to represent fractional numbers. So it is much more efficient for code to use a different (and simpler) int system for whole numbers.

▼ Our budget tracker program enables you to enter various data types: strings, ints, and lists

```
$ type(foo)
$ type(bar)
```

… and Shell will return:

```
<class 'int'>
<class 'float'>
```

So what happens if we add an int and float together? The answer is: it depends. Enter `type(foo + bar)` and you'll currently get "type<float>" because it's adding together a full number and a fraction. You can see the result by using `foo + bar` which returns: "46.2".

In general, if you add together ints you get an int, whereas if you add an int and a float together you get a float.

Type casting

It is important to change data types from one to another in Python through a process called 'type casting' (not to be confused with typecasting of actors). This makes it easy to turn an int into a float, or a string, or a different type. This is important because different functions may expect a specific data type (printing out an int requires it to be turned into a string first). You convert a float into an int using the `int()` function, placing the variable to be converted inside the brackets:

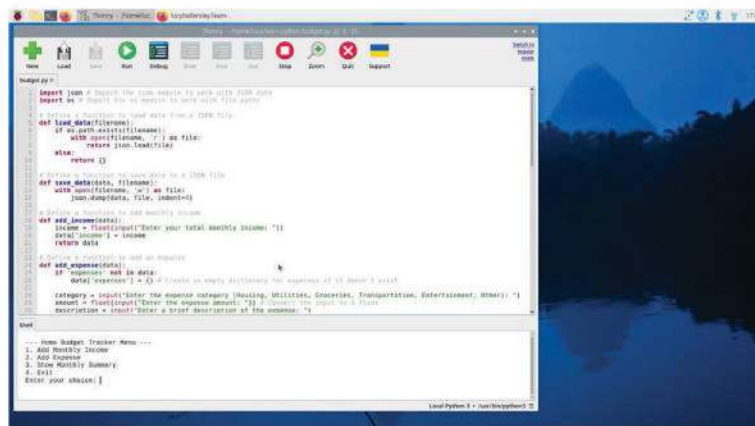```
$ foo = int(5.6)
$ foo
```

… returns 5. This is because 5.6, the value,

has been cast into an int and stored in `foo`. The process of turning a float into an int gets rid of the fractional value (it doesn't round up or down it gets rid of the fractional end: 5.9 becomes 5 and so on). The reverse process adds a ".0" to the end of an int.

```
$ bar = string(4)
$ bar
```

… returns a float: "4.0". One of the main uses for type casting is converting ints into strings so they can be printed out alongside messages. It's fine to enter `print(foo)` for example, which returns. But enter `print("The number is: " + foo)` and you'll get an error: "TypeError: can only concatenate str (not "int" to str". Reading this error carefully tells us that the error is a type error: our "int" cannot be concatenated (a fancy

way of saying "joined") to our string. We can fix this by type-casting the int to a string.

```
$ foo = 42
$ print("The number is: " + foo)
```

Which outputs "The number is: 42".

Strings

This brings us neatly to our second basic data type, the string. Strings are sequences of characters. They are typically used to represent words, but at a Python level they are a list of characters (including spaces). You create one just like an integer but you tell Python it's a string by enclosing it in quotation marks (single or double quotes):

```
$ string = "Hello!"
$ string
```

If you try entering a string without quotes, Python assumes that you're assigning one variable to another. Entering `string = Hello!` would produce an error (in this case "Invalid Syntax" as "!" is not a valid character for a variable name.

Strings can be added together, like numbers using the "+" operator, which combines:

```
$ greeting = "Hello"
$ name = "Alice"
$ message = greeting + ", " + name + "!"  #
"Hello, Alice!"
```

Enter message in Shell to view "Hello, Alice!"Strings are indexed characters, and can be accessed by adding square brackets "[]" afterwards with a number (starting at 0).

Test this out in Shell:

▼ The output of our budget tracker calculates your income and outgoings


```
$ string = "Hello, World!"
$ first_char = string[0]    # 'H'
$ last_char = string[-1]    # '!'
$ substring = string[7:12]  # 'World'
```

There's a lot you can do with strings, enough for its own chapter. A lot of Python is spent connecting, adding, slicing and dicing lists and strings. We'll come to Lists (which are like strings in a moment, first let's talk about True or False values.

True or False

A third type of basic data in Python is the boolean Value. Known as True or False. These values are entered without quotation marks, and with a capital "T" or "F".

```
$ a = True
$ b = False
```

True and False values are created throughout Python when you evaluate expressions or use logical operators.

```
$ a = (5 > 3)  # True
$ b = (5 == 3)  # False
$ print(a and b)  # False
$ print(a or b)  # True
$ print(not a)  # False
```

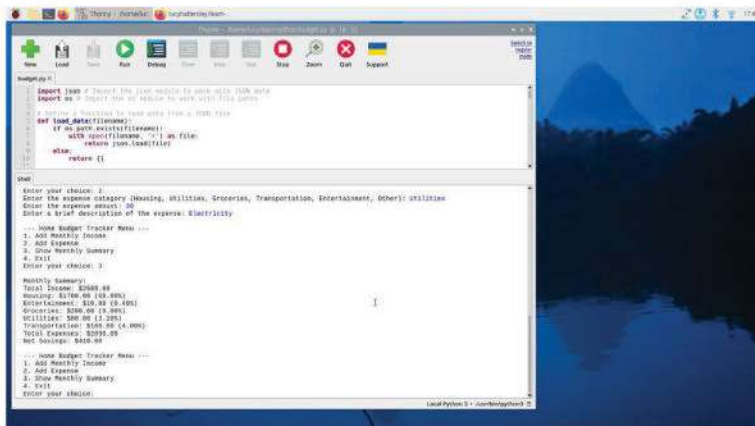You use True and False values in most Python programs to control flow.

```
$ a = True
$ if a:
    print("a is True")
$ else:
    print("a is False")
```
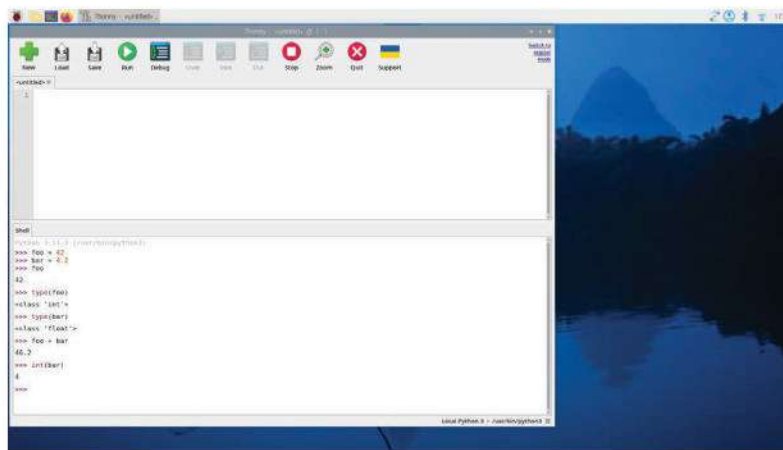
There's a concept known as "truthiness" in Python where various objects can be evaluated as true or false. Try this:

```
$ if "Hello":
    print("truthy")
$ else:
    print("not truthy")
```

Run this code and Python will print "truthy". Replace "Hello" with various things: 0, False, None, empty strings "" or lists [] will all run the else path and return "not truthy".

▶ The SHELL in Thonny IDE is useful for quickly entering Python commands and examining code

Positive or negative ints (–1, 1, 2, 3, and so on) evaluate to True. As do lists with items, strings with characters and so on. Experiment with different values to discover if they are inherently True or False.

Complex data types

Now that we've looked at the basic data types. It's time to have a look at more complex things. And the first item you'll encounter is a list.

Lists are created by using square brackets and placing items inside, with a comma separating each item. Lists can contain just about anything: ints, floats, strings, bools, even variables and other lists.

```
$ # Creating a list
$ my_list = [1, 2, 3, 4, 5]

$ # Lists can contain different data types
$ mixed_list = [1, "Hello", 3.14, True]
```

Items in a list may be accessed using square brackets with a number pointing to its position in the list (zero-indexed so the first item is 0, the second item is 1, and so on).

```
$ # Accessing elements
$ print(my_list[0])  # Output: 1
$ print(mixed_list[1])  # Output: "Hello"
```

The key thing with lists is that they are "mutable". That means you can change them: add, remove, swap values etc. This is typically done using dot notation commands after your variable. So to add an item you would use `my_list.append()` and put the new value inside the brackets. Here are some examples:

```
$ # Initial list
$ my_list = [1, 2, 3, 4, 5]  # my_list is [1,
2, 3, 4, 5]

$ # Appending items
$ my_list.append(6)  # my_list is now [1, 2,
3, 4, 5, 6]
```

```
$ # Inserting items
$ my_list.insert(2, "new")  # my_list is now
[1, 2, 'new', 3, 4, 5, 6]

$ # Removing items
$ my_list.remove("new")  # my_list is now [1,
2, 3, 4, 5, 6]

$ popped_item = my_list.pop(2)  # popped_item
is 3, my_list is now [1, 2, 4, 5, 6]

$ # Slicing lists
$ sliced_list = my_list[1:4]  # sliced_list
is [2, 4, 5]
```

What's a tuple?

A tuple is similar to a list, except you create it using round brackets instead of square ones. The big difference is that tuples are immutable, meaning that they cannot be changed after you have created them (you can't add, remove or reorder items in a tuple).

```
$ # Creating a tuple
$ my_tuple = (1, 2, 3)

$ # Tuples can contain different data types
$ mixed_tuple = (1, "Hello", 3.14, True)

$ # Single element tuple (note the comma)
```
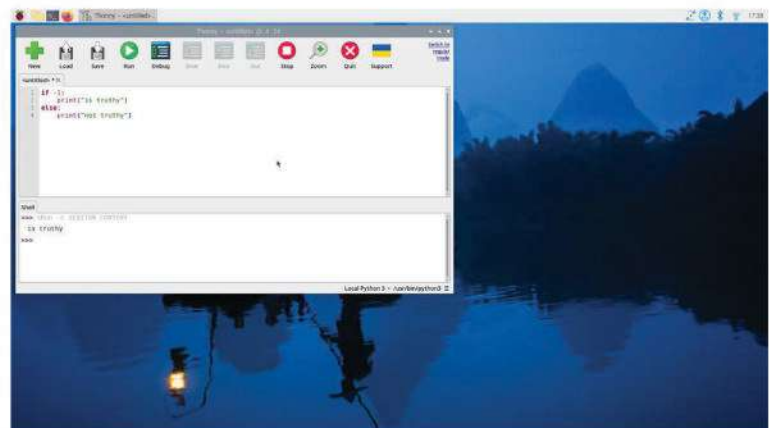
Top Tip 👍

Remember to cap

Remember to use capital "True" and "False". If you use lowercase letters Python will think you're creating variables called "true" and "false".

Top Tip 👍

Why 0?

Some programming languages, like R, index from 1, but Python, C, Java and most languages start at 0. Partly it's traditional (and soon becomes habit) and on a physical level items are stored in a RAM address and you can reference the first item, then add 1, 2, 3 and so on to get the next items in the list.

▼ Practice working with data in Thonny IDE to gain an understanding of how various Python elements work

```
$ single_element_tuple = (1,)
```

They have advantages over lists in that they can be fixed.

Dictionaries

A dictionary (also known as a "dict") is a list with key/value pairs. That means you add a key (which can be a word such as "name", and "age" and a value such as "Bob" and "32".

Then, when accessing the values in a dictionary you use the key rather than the index position (as you would in a list). So rather than saying `item[0]` and getting "Bob" you would say `item["name"]`. Let's create and use a dictionary:

```
$ # Creating a dictionary
$ my_dict = {"name": "Alice", "age": 25,
"city": "Cambridge"}

$ # Accessing values
$ print(my_dict["name"])  # Output: Alice
```

Like lists, you can add values. You do this using square brackets with the new key, an equals assignment operator and the value:

```
$ # Adding or modifying values
$ my_dict["age"] = 26  # Modifying an
existing value
$ my_dict["email"] = "alice@example.com"  #
Adding a new key-value pair
```

You can delete values using `del` and the key, but it's more common to use a command called `pop`. This 'pops' a value off the list and returns it to be stored.

```
$ # Using del
$ del my_dict["city"]

$ # Using pop()
$ email = my_dict.pop("email")
```

Don't be surprised if key/value pairs are confusing at first. It takes a while!

Wrapping up

That's our whistle-stop tour of Python data types wrapped up. Now enter our **budget.py** code to create a working budget tracker. It also saves data to a JSON (JavaScript Object Notation) file. You can learn more about JSON at **json.org**. 🔲

budget.py

▶ Language: **Python**

```python
001.  import json # Import the json module to work
      with JSON data
002.  import os # Import the os module to work with
      file paths
003.
004.  # Define a function to load data from a JSON file
005.  def load_data(filename):
006.      if os.path.exists(filename):
007.          with open(filename, 'r') as file:
008.              return json.load(file)
009.      else:
010.          return {}
011.
012.  # Define a function to save data to a JSON file
013.  def save_data(data, filename):
014.      with open(filename, 'w') as file:
015.          json.dump(data, file, indent=4)
016.
017.  # Define a function to add monthly income
018.  def add_income(data):
019.      income = float(input(
      "Enter your total monthly income: "))
020.      data['income'] = income
021.      return data
022.
023.  # Define a function to add an expense
024.  def add_expense(data):
025.      if 'expenses' not in data:
026.          data['expenses'] = {} # Create an empty
      dictionary for expenses if it doesn't exist
027.
028.      category = input("Enter the expense category
      (Housing, Utilities, Groceries, Transportation,
      Entertainment, Other): ")
029.      amount = float(input("Enter the expense
      amount: ")) # Convert the input to a float
```

```
030.      description = input(
      "Enter a brief description of the expense: ")
031.
032.      if category in data['expenses']:
033.          data['expenses'][category].
034. append({'amount': amount,
      'description': description})
035.      else:
036.          data['expenses'][category] =
      [{'amount': amount, 'description': description}]
037.
038.      return data
039.
040. # Define a function to display a monthly summary
041. def monthly_summary(data):
042.      total_expenses = 0 # Initialize a variable
      to store the total expenses
043.      income = data.get('income', 0)
044.      expenses = data.get('expenses', {})
045.
046.      print("\nMonthly Summary:")
047.      print(f"Total Income: ${income:.2f}")
048.
049.      for category, entries in expenses.items():
050.          category_total = sum(
      item['amount'] for item in entries)
051.          total_expenses += category_total
052.          print(f"{category}: ${category_
      total:.2f} ({(category_total / income *
      100):.2f}%)")
053.
054.      net_savings = income - total_expenses #
      Calculate the net savings
055.      print(
      f"Total Expenses: ${total_expenses:.2f}")
056.      print(f"Net Savings: ${net_savings:.2f}")
057.
058. # Define the main function
059. def main():
060.      data_file = 'budget_data.json'
061.      budget_data = load_data(data_file)
062.
063.      while True:
064.          print(
      "\n--- Home Budget Tracker Menu ---")
065.          print("1. Add Monthly Income")
066.          print("2. Add Expense")
067.          print("3. Show Monthly Summary")
068.          print("4. Exit")
069.          choice = input("Enter your choice: ")
070.
071.          if choice == '1':
072.              budget_data =
      add_income(budget_data)
073.          elif choice == '2':
074.              budget_data =
      add_expense(budget_data)
075.          elif choice == '3':
076.              monthly_summary(budget_data)
077.          elif choice == '4':
078.              save_data(budget_data, data_file)
079.              print(
      "Data saved. Exiting program.")
080.              break
081.          else:
082.              print(
      "Invalid choice. Please choose a valid option.")
083.
084. # Call the main function
085. if __name__ == "__main__":
086.      main()
```

Part 04

Programming with
Scratch 3

Learn how to start coding using Scratch, a block-based programming language.

MAKER

Gareth Halfacree

Gareth is a freelance technology journalist, writer, and former system administrator in the education sector with a passion for open-source software and hardware.

freelance. halfacree.co.uk

Key to the accessibility of coding on Raspberry Pi is Scratch, a visual programming language developed by the Massachusetts Institute of Technology (MIT). Rather than using text-based instructions, Scratch has you build your program using blocks – pre-written chunks of code disguised as colour-coded jigsaw pieces.

Scratch is a great first language for budding coders of any age, and you can use it to create everything from simple games and animations through to complex interactive robotics projects.

Scratch 3 loads like any other program on Raspberry Pi: open the Raspberry Pi menu, move the cursor to the Programming section, and click on Scratch 3. After a few seconds, the Scratch 3 user interface will display. You may see a message about data collection: you can click Yes if you're happy to submit usage data, otherwise click No.

Start by clicking on the Looks category in the blocks palette, found at the left of the Scratch window. This brings up the purple blocks under that category. Find the "Say Hello" block, click and hold the left mouse button on it, and drag it over to the code area at the centre of the Scratch window before letting go of the mouse button (**Figure 1**).

Say hello to the Scratch interface

Look at the shape of the block you've just dropped: it has a hole at the top and a matching part sticking out at the bottom. Like a jigsaw piece, this shows you that the block is expecting to have something above it and something below it. For this program, that something above is a trigger.

Click on the Events category of the blocks palette, coloured gold, then click and drag the `when green flag clicked` block – known as a hat block – onto the code area. Position it so that the bit sticking out of the bottom connects into the hole at the top of your `say Hello` block until you see a white outline,

Top Tip 👍

Scratch versions

There are two versions of Scratch available for Raspberry Pi OS: Scratch and Scratch 3. This article is written with Scratch 3 in mind, which is only compatible with Raspberry Pi 4, Raspberry Pi 5, and Raspberry Pi 400.
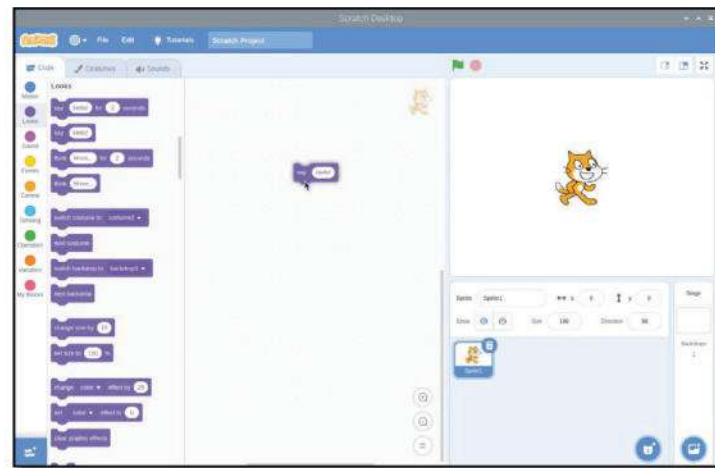
▲ **Figure 1:** Drag and drop the block into the code area

then let go of the mouse button. You don't have to be precise; if it's close enough, the block will snap into place. If it doesn't, click and hold on it again to adjust its position until it does (see **Figure 2**).

Your program is now complete. To make it work, known as running the program, click the green flag icon at the top-left of the stage area. If all has gone

> ❝ The block is expecting to have something above it and something below it ❞

well, the cat sprite on the stage will greet you with a cheery 'Hello!' – your first program is a success!

Before moving on, name and save your program. Click on the File menu, then Save to your computer. Type in a name and click the Save button (**Figure 3**).

Your program has two blocks, but it only has one real instruction: to say 'Hello!' every time the
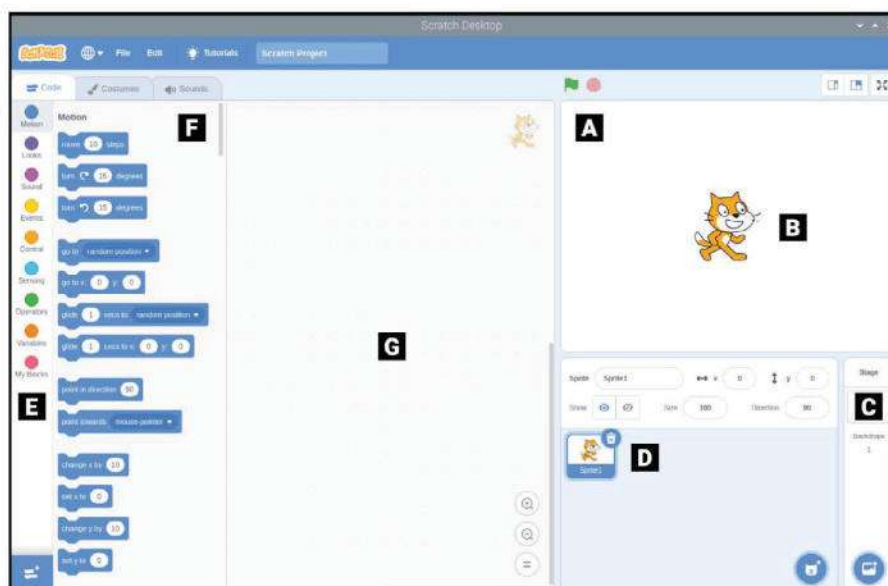
A. **Stage area.** Like actors in a play, your characters move around the stage under the control of your Scratch program

B. **Sprite.** These characters are known as sprites

C. **Stage controls.** To change the stage, such as to add your own background, use the stage controls

D. **Sprites list.** All the sprites you have created or loaded are in the Sprites list

E. **Blocks palette.** All the blocks available for your program appear in the blocks palette, which features colour-coded categories

F. **Blocks.** Blocks are pre-written chunks of code

G. **Code area.** You build your program in the code area by dragging and dropping blocks from the blocks palette to form scripts
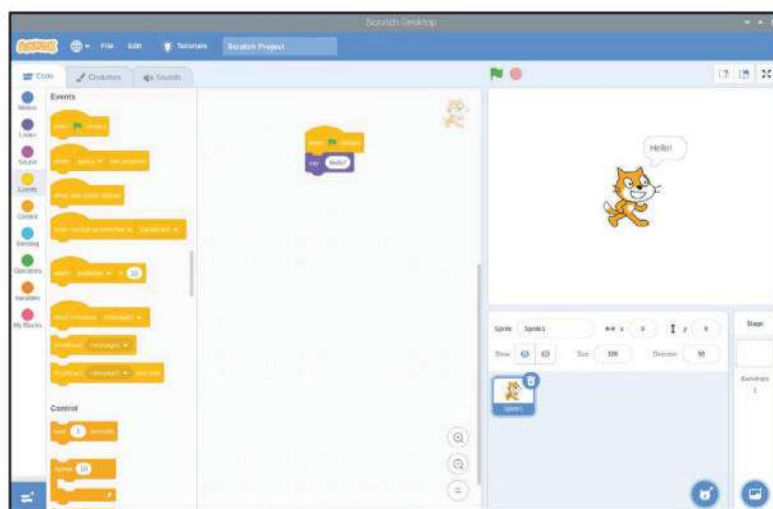
program runs. To do more, you need to know about sequencing. Computer programs, at their simplest, are a list of instructions, just like a recipe. Each instruction follows on from the last in a logical progression known as a linear sequence.

Next steps

Start by clicking and dragging the `say Hello!` block from the code area back to the blocks palette. This deletes the block, removing it from your program and leaving just the Trigger block, `when green flag clicked` (**Figure 4**).

Click on the Motion category in the blocks palette, then click and drag the `move (10) steps` block so it locks into place under the trigger block on the code area.

As the name suggests, this tells your sprite – the cat – to move a set number of steps in the direction it's currently facing.

Next, add more instructions to your program to create a sequence: click on the Sound category, colour-coded pink, then click and drag the `play sound [Meow] until done` block so it locks underneath the `move (10) steps` block. Now keep the sequence going: click on the Motion category
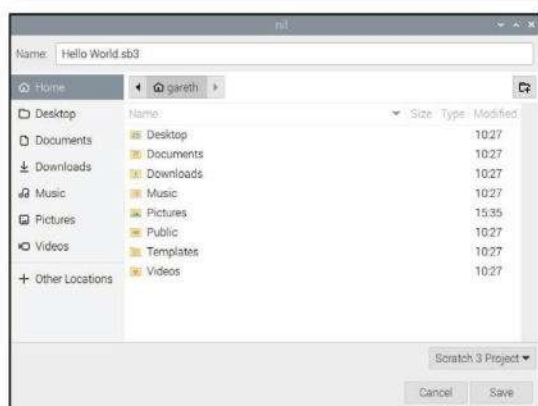
▲ **Figure 3:** Save your program with a memorable name

◀ **Figure 2:** The when green flag blog clicked into say [Hello!]. Click the green flag above the stage and the cat will say 'Hello'

You'll Need

> Raspberry Pi (4, 400, 5)

> Raspberry Pi OS

> Internet connection
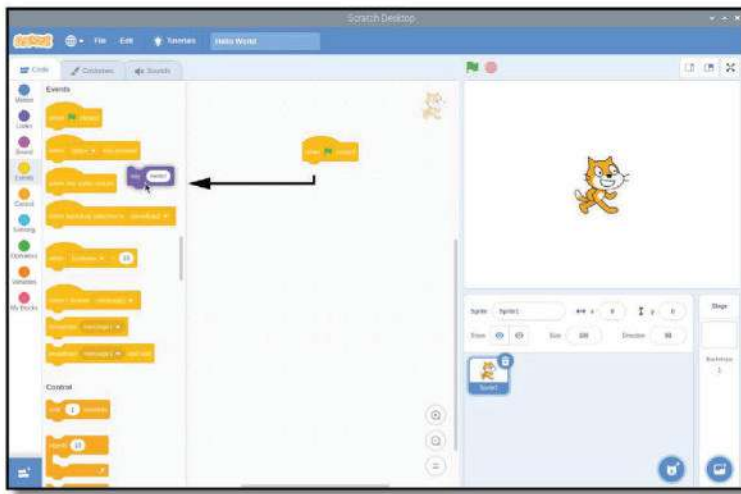
▲ **Figure 4:** To delete a block, simply drag it out of the code area
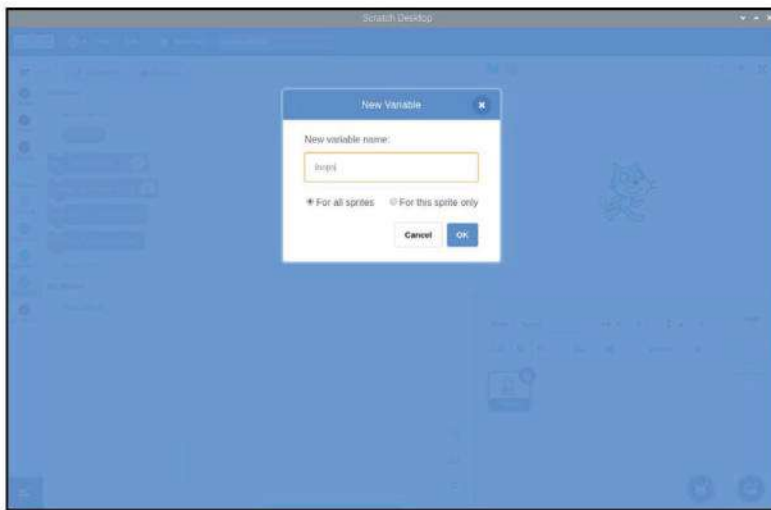
Top Tip 👍

What can it say?

Some blocks in Scratch can be modified. Try clicking on the word 'Hello!', then type something else and click the green flag again. What happens on the stage?

Top Tip 👍

Add steps

Try adding more steps to your sequence and changing the values in the existing steps. What happens when the number of steps in one move block doesn't match the number of steps in another? What happens if you try to play a sound while another sound is still playing?

again and drag another `move (10) steps` block underneath your Sound block, but this time change "10" to "–10" to create a `move (-10) steps` block.

```
when green flag clicked
move (10) steps
play sound [Meow] until done
move (-10) steps
```

Click the green flag above the stage to run the program. You'll see the cat move to the right, make a 'meow' sound – make sure you've got speakers or headphones connected to hear it – then move back to the start again. Every time you click the green flag the cat will repeat these actions.

Congratulations! You've created a sequence of instructions, which Scratch is running through one at a time, top to bottom. While Scratch will only run one instruction at a time from the sequence, it does so very quickly: try deleting the `play sound [Meow] until done` block by clicking and dragging the bottom `move (-10) steps` block to detach it, dragging the `play sound [Meow] until done` block to the blocks palette, then replacing it with the simpler `play sound [Meow]` block before dragging your `move (-10) steps` block back onto the bottom of your program.

```
when green flag clicked
move (10) steps
play sound [Meow]
move (-10) steps
```

Click the green flag to run your program again. Only this time the cat sprite doesn't seem to move. The sprite is moving, but it moves back again so quickly that it appears to be standing still. This is because using the `play sound [Meow]` block doesn't mean that the program will wait for the sound to finish playing before the next step. Because Raspberry Pi 'thinks' so quickly, the next instruction

runs before you can see the cat sprite move. There's another way to fix this, beyond using the `play sound [Meow] until done` block: click on the light orange Control category of the blocks palette, then click and drag a `wait (1) seconds` block between the `play sound [Meow` block and the bottom `move (-10) steps` block.

```
when green flag clicked
move (10) steps
play sound [Meow v]
wait (1) seconds
move (-10) steps
```

Click the green flag to run your program one last time, and you'll see that the cat sprite waits for a second after moving to the right before moving back again. This is known as a delay, and is key to controlling how long your sequence of instructions takes to run.

Looping the loop

The sequence you've created so far runs only once. You click the green flag, the cat sprite moves and meows, and then the program stops until you click the green flag again. It doesn't have to stop, though, because Scratch includes a type of Control block known as a loop.

Click on the Control category in the blocks palette and find the `forever` block. Click and drag this into the code area, then drop it underneath the `when green flag clicked` block and above the first `move (10) steps` block.

```
when green flag clicked
forever
move (10) steps
play sound [Meow v]
wait (1) seconds
move (-10) steps
```

The C-shaped `forever` block automatically grows to surround the other blocks in your sequence. Click the green flag now and you'll quickly see what the `forever` block does: instead of your program running once and finishing, it will run over and over again – quite literally forever. In programming, this is known as an infinite loop – a loop that never ends.

If the sound of constant meowing is getting to be a little much, click the red octagon next to the green flag above the stage area to stop your program. To change the loop type, pull the first `move (10) steps` block, and the blocks beneath it, out of the `forever` block, then drop them

underneath the `when green flag clicked` block. Click and drag the `forever` block to the blocks palette to delete it, then click and drag the `repeat (10)` block under the `when green flag clicked` block so it goes around the other blocks.

```
when green flag clicked
repeat (10)
move (10) steps
play sound [Meow v]
wait (1) seconds
move (-10) steps
```

Click the green flag to run your new program. At first, it seems to be doing the same thing as your original version: repeating your sequence of instructions over and over again. This time, though, rather than continuing forever, the loop will finish after ten repetitions. This is known as a definite loop – you define when it will finish. Loops are powerful tools, and most programs, especially games and sensing programs, make heavy use of both infinite and definite loops.

Variables and conditionals

The final concepts you'll need to understand before beginning to code Scratch programs in earnest are closely related: variables and conditionals. A variable is, as the name suggests, a value which can vary – in other words, change – over time and under control of the program. A variable has two main properties: its name, and the value it stores. That value doesn't have to be a number, either. It can be numbers, text, true-or-false (also known as boolean values), or completely empty – known as a null value.

Variables are powerful tools. Think of the things you have to track in a game: the health of a character, the speed of moving object, the level being played, and the score. All of these are tracked as variables.

First, click the File menu and save your existing program by clicking Save to your computer. If you saved the program earlier, you'll be asked if you want to overwrite it, replacing the old saved copy with your new, up-to-date version. Next, click File and then New to start a new, blank project (click OK when asked if you want to replace the contents of the current project).

Click on the dark orange Variables category in the blocks palette, then the Make a Variable button. Type "loops" as the variable name, then click OK to make a series of blocks appear in the blocks palette (**Figure 5**).

Click and drag the `set loops to [0]` block to the code area. This tells your program to initialise the variable with a value of 0. Next, click on the Looks category of the blocks palette and drag the `say [Hello!] for (2) seconds` block under your `set loops to [0]` block.

```
set [loops v] to (0)
say [Hello!] for (2) seconds
```

As you found earlier, the `say [Hello!]` blocks cause the cat sprite to say whatever is written in them. Rather than writing the message in the block yourself, though, you can use a variable instead. Click back onto the Variables category in the blocks palette, then click and drag the rounded (`loops`) block – known as a reporter block, found at the top of the list with a tick-box next to it – over the word Hello! in your `say [Hello!] for (2) seconds` block. This creates a new, combined block: `say [loops] for (2) seconds`.

```
set [loops v] to (0)
say (loops) for (2) seconds
```

Click on the Events category in the blocks palette, then click and drag the `when green flag clicked` block to place it on top of your sequence of blocks. Click the green flag above the stage area, and you'll see the cat sprite say '0' – the value you gave to the variable 'loops' (**Figure 6**).

Variables aren't unchanging, though. Click on the Variables category in the blocks palette, then click and drag the `change loops by 1` block to the bottom of your sequence.

Next, click on the Control category, then click

Top Tip 👍

What happens now?

What happens if you change the number in the loop block to make it larger? What happens if it's smaller? What happens if you put the number 0 in the loop block?
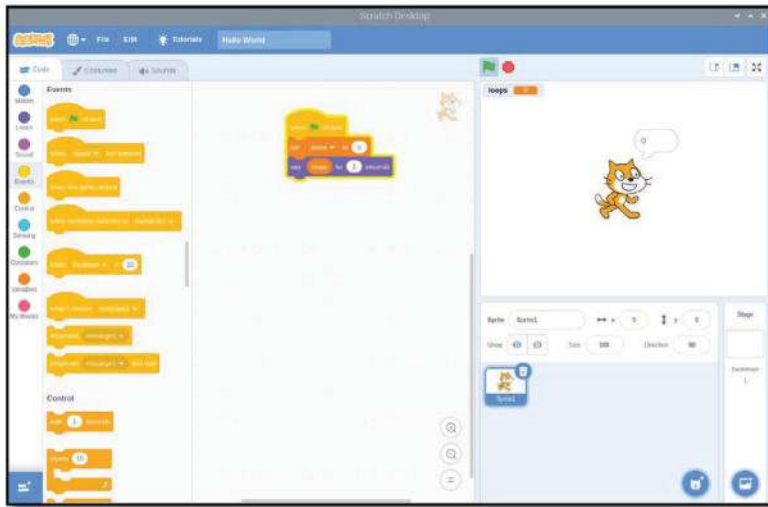
▲ **Figure 6:** This time the cat will say the value of the variable

and drag a `repeat (10)` and drop it so that it starts directly beneath your `set loops to 0` block and wraps around the remaining blocks in your sequence.
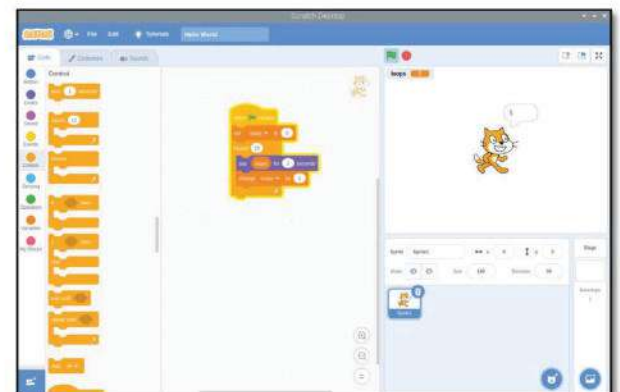
```
when green flag clicked
set [loops v] to (0)
repeat (10)
say (loops) for (2) seconds
change [loops v] by (1)
```

Click the green flag again. This time, you'll see the cat count upwards from 0 to 9. This works because your program is now changing, or modifying, the variable itself: every time the loop runs, the program adds one to the value in the 'loops' variable.

You can do more with a variable than modify it. Click and drag the `say (loops) for (2) seconds` block to break it out of the `repeat (10)` block and drop it below the `repeat (10)` block. Click and drag the `repeat (10)` block to the blocks palette to delete it, then replace it with a `repeat until` block, making sure the block is connected to the bottom of the set `loops to [0]` block. It should surround both of the other blocks in your sequence. Next, click the Operators category in the blocks palette, colour-coded green, then click and drag the diamond-shaped `() = ()` block and drop it on the matching diamond-shaped hole in the `repeat until::control` reporter block.

This Operators block lets you compare two values, including variables. Click on the Variables category, drag the `(loops)` reporter block into the empty space in the `() = ()` Operators block, then click on the space with 50 in it and type the number 10.

When green flag clicked:

```
set [loops v] to (0)
repeat until <(loops) = (10)>
say (loops) for (2) seconds
```

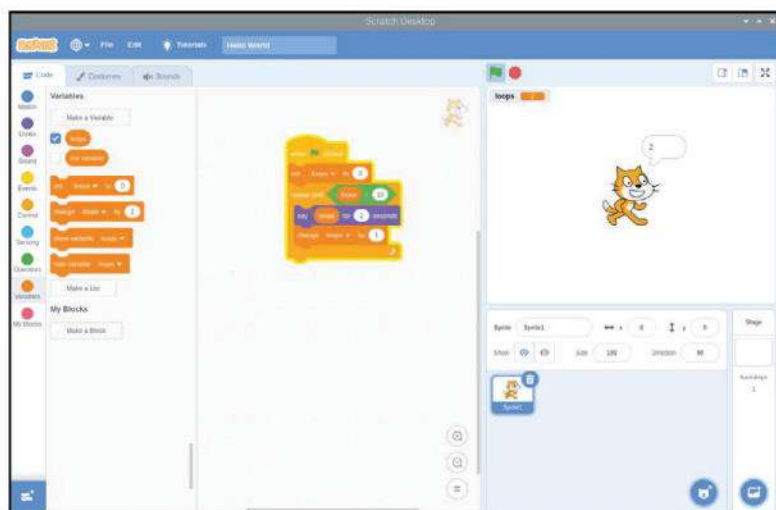▶ **Figure 7:** Thanks to the loop, the cat now counts upwards

```
change [loops v] by (1)
```

Click on the green flag above the stage area and you'll find the program works the same way as before: the cat sprite counts from 0 up to 9 and then the program stops. This is because the repeat until block is working in exactly the same way as the `repeat (10)` block, but rather than counting the number of loops itself, it's comparing the value of the loops variable to the value you typed to the right of the block. When the loops variable reaches 10, the program stops.

This is known as a comparative operator: it literally compares two values. Click on the Operators category of the blocks palette and find the two other diamond-shaped blocks above and below the one with the = symbol. These are also comparative operators: < compares two values and is triggered when the value on the left is smaller than the one on the right, and > triggers when the value on the left is larger than the one on the right.

Click on the Control category of the blocks palette, find the `if then` block, then click and drag it to the code area before dropping it directly beneath the `say [loops] for (2) seconds`. It will

> ## 💬 This Operators block lets you compare two values, including variables 💬

automatically surround the `change loops by 1` block, so click and drag on that block to move it so it connects to the bottom of your `if then` block instead. Click on the Looks category of the blocks palette, then click and drag a `say [Hello!] for (2) seconds` block and drop it inside your `if then` block. Click on the Operators category of the

blocks palette, then click and drag the `() > ()` block into the diamond-shaped hole in your `if then::control` reporter block.

The `if then` block is a Conditional block, which means the blocks inside it will only run if a certain condition is met. Click on the Variables category of the blocks palette, drag and drop the `(loops)` reporter block into the empty space in your `() > ()` block, then click on the space with 50 in it and type the number 5. Finally, click on the word Hello! in your `say [Hello!] for (2) seconds` block and type "That's high!".

```
when green flag clicked
set [loops v] to (0)
repeat until <(loops) = (10)>
say (loops) for (2) seconds
if <( loops) > (5)> then
say [That's High!] for (2) seconds
end
change [loops v] by (1)
```

Click the green flag to run your program. At first, it will work as before with the cat sprite counting upwards from zero. When the number reaches six, the first number greater than five, the `if then` block will begin to trigger and the cat sprite will comment on how high the numbers are getting (**Figure 9**). Congratulations: you can now work with variables and conditionals!

Build an astronaut reaction timer

Now that you understand how Scratch works, it's time to make something a little more interactive: a reaction timer, designed to honour British ESA astronaut Tim Peake and his time aboard the International Space Station.

Save your existing program, if you want to keep it, then open a new project by clicking on

File and New. Before you begin, give it a name by clicking on File and Save to your computer: call it 'Astronaut reaction timer'.

This project relies on two images – one as a stage background, one as a sprite – which are not included in Scratch's built-in resources. To download them, click on the Raspberry Pi icon to load the Raspberry Pi menu, move the mouse pointer to Internet, and click on Chromium Web Browser. When the browser has loaded, type **rptl.io/astro-bg** into the address bar, then press the **ENTER** key. Right-click on the picture of space and click on Save image as, then click on the Save button (**Figure 10**). Click back into the address bar, and type **rptl.io/astro-sprite** followed by the **ENTER** key.

Again, right-click on the picture of Tim Peake and click on Save image as, then choose the **Downloads** folder and click on the Save button. With those two images saved, you can close Chromium or leave it open and use the taskbar to switch back to Scratch 3.

Right-click the cat sprite in the list and click delete. Hover the mouse pointer over the Choose a Backdrop icon. Next, click the Upload Backdrop icon from the list that appears.
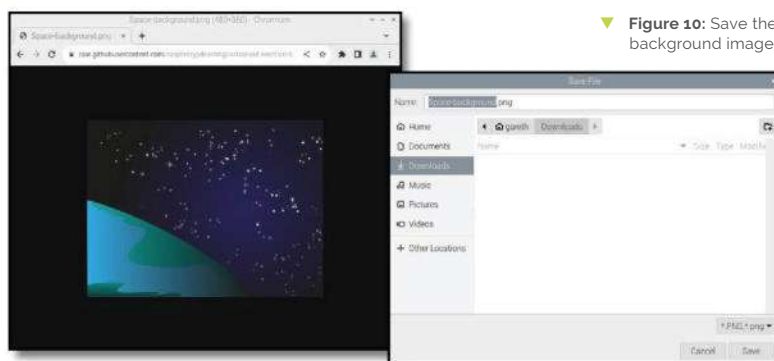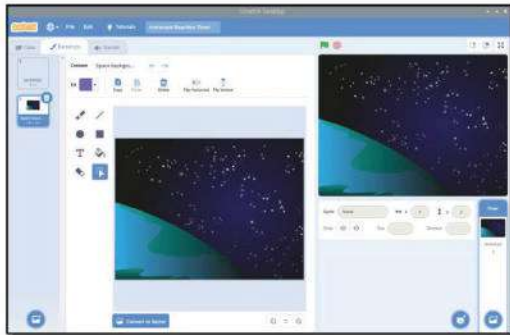
Figure 11: The space background appears on the stage

Figure 12: Drag the astronaut sprite to the lower middle of the stage

Find the **Space-background.png** file in the **Downloads** folder, click on it to select it, then click OK. The plain white stage background will change to the picture of space, and the code area will be replaced by the backdrops area (**Figure 11**). Here you can draw over the backdrop, but for now just click on the Code tab at the top of the Scratch 3 window.

Upload your new sprite by hovering your mouse pointer over the Choose a Sprite icon. Next, click the Upload Sprite icon at the top of the list that appears. Find the file **Astronaut-Tim.pn**g in the **Downloads** folder, click to select it, then click OK. The sprite appears on the stage automatically, but it might not be in the middle of the stage: click and drag it with the mouse and drop it so it's near the lower middle (**Figure 12**).

With your new background and sprite in place, you're ready to create your program, so click the Code tab. Start by creating a new variable called time, making sure that For all sprites is selected before clicking OK. Click on your sprite – either on the stage or in the sprite pane – to select it, then add a When green flag clicked :: reporter block from the Events category to the code area. Next, add a say [Hello!] for (2) seconds:: reporter block from the Looks category, then click on it to change it to say Hello! British ESA Astronaut Tim Peake here. Are you ready?

```
when green flag clicked
say [Hello! British ESA Astronaut Tim Peake
here. Are you ready?] for (2) seconds
```

Add a `wait (1) seconds` block from the Control category, then a `say [Hello!]` block. Change this block to say 'Hit Space!', then add a `reset timer` block from the Sensing category. This controls a special Scratch variable for timing things, and will be used to see how quickly you react.

```
when green flag clicked
say [Hello! British ESA Astronaut Tim Peake
here. Are you ready?] for (2) seconds
wait (1) seconds
say [Hit Space!]
reset timer
```

Add a `wait until` block, then drag a `key space pressed?` Sensing block into its white space. This will pause the program until you press the **SPACE** key on the keyboard, but the timer will continue to run – counting exactly how much time has passed between the message telling you to 'Hit Space!' and when you actually press the **SPACE** key.

```
when green flag clicked
say [Hello! British ESA Astronaut Tim Peake
here. Are you ready?] for (2) seconds
wait (1) seconds
say [Hit Space!]
reset timer
wait until <key [space v] pressed?>
```

You now need Tim to tell you how long you took to press the **SPACE** key, but in a way that's easy for you to read. To do this, you'll need a `join () ()` Operators block. This takes two values, including variables, and joins them together one after the other – known as concatenation.

Start with a `say [Hello!]` block, then drag and drop a `join () ()` Operators block over the word Hello!. Click on apple and type "Your reaction time was " – make sure you have a blank space at the end – then drag another `Join` block over the top of banana in the second box. Drag a `timer` Reporting block from the Sensing category into what is now the middle box, and type " seconds." into the last box – make sure to include a blank space at the start.

```
when green flag clicked
say [Hello! British ESA Astronaut Tim Peake
here. Are you ready?] for (2) seconds
wait (1) seconds
say [Hit Space!]
reset timer
wait until <key [space v] pressed?>
say (join [Your reaction time was  ] (join
(timer) [ seconds])
```

Finally, drag a `set my variable to [0]` Variables block onto the end of your sequence. Click on the drop-down arrow next to 'my variable' and click
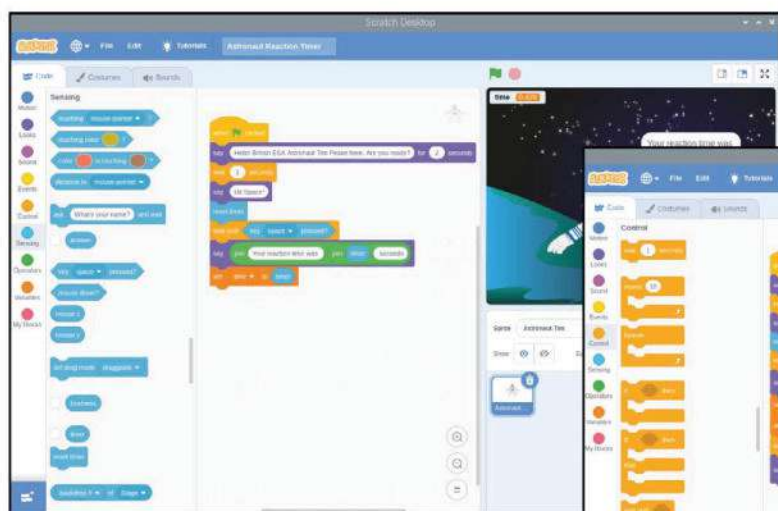
on 'time' from the list, then replace the 0 with a timer Reporting block from the Sensing category. Your game is now ready to test by clicking on the green flag. As soon as you see 'Hit Space!', press the **SPACE** key as quickly as you can (**Figure 13**).

You can extend this project further by having it calculate roughly how far the International Space Station has travelled in the time it took you to press the **SPACE** key, based on the station's published speed of 7km/s. First, create a new variable called distance. Notice how the blocks in the Variables category automatically change to show the new variable, but the existing (time) variable blocks in your program remain the same.

Add a `set distance to [0]` block, then drag a `() * ()` Operators block – which indicates multiplication – over the 0. Drag a (time) Reporting block over the first blank space, then type the number 7 in the second space. When you're finished, your combined block reads `set distance to time * (7)::variables reporter`. This will take the time it took you to press the **SPACE** key and multiply it by seven, to get the distance in kilometres the ISS has travelled.

```
when green flag clicked
say [Hello! British ESA Astronaut Tim Peake
here. Are you ready?] for (2) seconds
wait (1) seconds
say [Hit Space!]
reset timer
wait until <key [space v] pressed?>
say (join [Your reaction time was  ] (join
(timer) [ seconds])
set [time v] to (timer)
set [distance v] to ((time) * (7))
```

Add a `wait (1) seconds` block and change it to 4. Next, drag another `say [Hello!]::` reporter
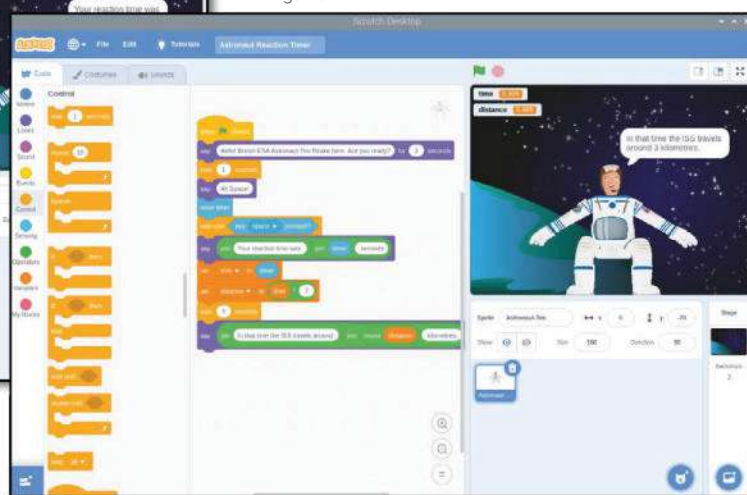
block onto the end of your sequence and add two `join::operators` reporter blocks, just as you did before. In the first space, over apple, type "In that time the ISS travels around ", remembering a space at the end; in the banana space, type " kilometres", again remembering a space at the start.

Finally, drag a `join` Operators block into the middle blank space, then drag a `distance` Reporting block into the new blank space it creates. The `join` block rounds numbers up or down to their nearest whole number, so instead of a hyper-accurate but hard-to-read number of kilometres you'll get an easy-to-read whole number.

Click the green flag to run your program and see how far the ISS travels in the time it takes you to hit the **SPACE** key. Remember to save your program when you're finished, so you can easily load it again in the future without having to start from the beginning (**Figure 14**)! ▣

```
when green flag clicked
say [Hello! British ESA Astronaut Tim Peake
here. Are you ready?] for (2) seconds
wait (1) seconds
say [Hit Space!]
reset timer
wait until <key [space v] pressed?>
say (join [Your reaction time was  ] (join
(timer) [ seconds])
set [time v] to (timer)
set [distance v] to ((time) * (7))
wait (4) seconds
say (join [In that time the ISS travels
around  ] (join (distance:: variables) [
kilometres.])
```

Top Tip 👍

Challenge: custom artwork

You can click on a sprite or background, then click the Costumes or Backdrops tab to bring up an editor with drawing tools. Can you draw your own characters and background and edit the code to have your character say something different?

CDP Studio:
Control a robot arm with a Wii Remote

Part 02

Use CDP Studio with a Nintendo Wii controller to manipulate a Raspberry Pi-based robot arm

MAKER

Phil King

Long-time contributor to *The MagPi*, Phil is a freelance writer and editor with a focus on technology.

@philkingeditor

Last issue, we showed you how to use the CDP Studio software development tool and its Kinematics framework to record movements for a Raspberry Pi-based robotic arm – the myCobot 280 Pi from Elephant Robotics – that could then be played back to perform a 'pick and place' routine using the Adaptive Gripper add-on. This time we'll using the same setup, but controlling the arm manually using a Nintendo Wii Remote and Nunchuk.

If you don't have the robot arm, you can still run the project by deploying it to a Raspberry Pi and viewing the movements in the on-screen arm visualiser.

▼ You need to pair CDP Studio with the arm's Raspberry Pi over the Wi-Fi network to deploy the project

01 Install the software

If you haven't already done it for the previous tutorial, you'll need to install CDP Studio. On your PC, visit **cdpstudio.com/getstarted** and download the free non-commercial version for Windows or Linux. During installation, select the 'ARMv8 64-bit (Debian 11)' component under

CDP Studio 4.12, along with the one already ticked for your host PC. You will then be able to deploy projects to the myCobot 280 Pi arm, which uses a 64-bit version of Ubuntu.

If you already have CDP Studio installed, make sure it's updated to the latest version (Help > Check For Updates). Then go to Help > Package Manager and select 'Add or remove CDP versions' to add the ARMv8 64-bit (Debian 11) component if not already added.

02 Download the project

This is a complex project that would be time-consuming to build from scratch, so we'll download it from CDP Studio's GitHub repo. Go to **magpi.cc/mycobotwiiremote**, click the green Code button, and select 'Download ZIP'. Unzip the file on your PC. Move the resulting **myCobotWiiRemote-main** folder to the **CDPStudioWorkspace/systems** folder.

03 Download and build library

To deploy the project to the robot arm, you'll need a couple of libraries. If you don't already have the myCobotLib library from the previous tutorial, go to the GitHub repo at **magpi.cc/mycobotlib**, click the Code button and Download ZIP again. Extract it and then move the resulting **myCobotLib-main** folder to the **CDPStudioWorkspace/libraries** folder.

Open the myCobotLib.pro project file in CDP Studio, check ARM 64-bit is checked in the Deploy Configuration tab, then right-click its name in the left panel and select Build.

You'll also need the xwiiremotelib library at **magpi.cc/xwiimotelib** and the xwiimote
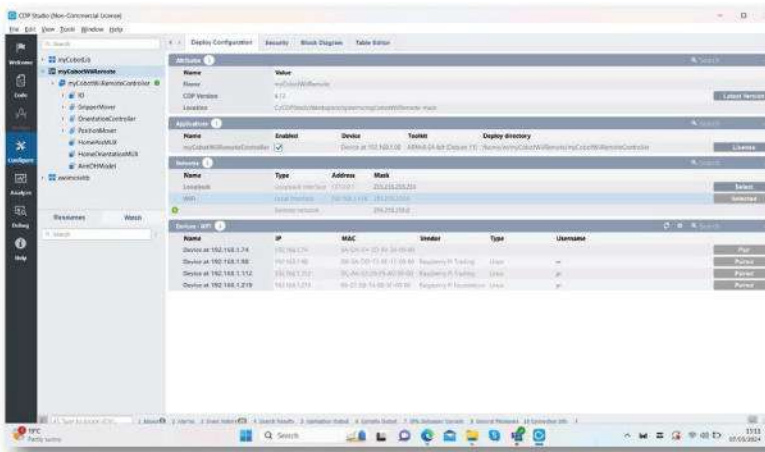
We run the project in CDP Studio on a PC and deploy it over the network to the myCobot's Raspberry Pi

The arm's end-point can be moved and its Adaptive Gripper attachment opened and closed

The myCobot 280 Pi robot arm will follow the movements of the Wii Remote or Nunchuk

You'll Need

> Windows or Linux PC

> CDP Studio 4.12
  **cdpstudio.com/
  getstarted**

> myCobot 280 Pi
  robot arm
  **magpi.cc/
  mycobot280**

> myCobot
  Adaptive Gripper
  **magpi.cc/
  mycobotgripper**

> Wii Remote
  and Nunchuk

submodule on which it's based at **magpi.cc/ xwiimote**. You can either clone the library with the recursive option:

```
$ git clone --recursive https://github.com/
CDPTechnologies/xwiimotelib.git
```

Or you can download and extract the ZIP file, then download and extract the xwiimote ZIP and place the folder's contents into the library's xwiimote subfolder.

Move the **xwiimotelib-main** folder to **CDPStudioWorkspace/libraries**, then open the xwiiremotelib.pro project file in CDP Studio. Before building the library, open the Deploy Configuration tab for the wiimoteIO component and check that the ARMv8 64-bit (Debian 11) toolkit is enabled. If you get errors when building it, make sure CDP Studio has been updated (see Step 1).

04 Open the project

Now open the main **myCobotWiiRemote. pro** project file in CDP Studio. In the left panel of the Configure window, you'll note that it

▲ Pair and then connect the Wii Remote controller with the arm using the Bluetooth Manager in Ubuntu

comprises a myCobotWiiRemoteController application with various components. A key part is the Kinematics-based ArmDHModel, which features a DHChain Visualizer pane that can be used to view the arm positions on-screen in 3D. This can be used even if you don't have a real arm connected, so you can still run the project and see how your Wii Remote and Nunchuk actions affect its movements.

▲ You can test the Wii Remote with the 'xwiishow 1' command in a terminal in the arm's Ubuntu OS

Top Tip 👍

Another arm

While this project is designed for the myCobot 280 Pi, you could use a different robot arm – you'd just need to create a new IO library to communicate with it.

05 Prepare myCobot

If you've already set this up in the first tutorial, you can skip the first part of this step. The myCobot arm's Ubuntu OS has a non-standard version of the OpenSSH server. So you'll need to make a small change to a config file so CDP Studio can communicate with it over the network. SSH into the myCobot with the username 'er' at its IP address; the default password is Elephant. Then enter:

```
$ sudo nano /etc/ssh/sshd_config
```

Locate the line that sets the PubkeyAuthentication parameter and set it to yes (and make sure the line is not commented out). Press **CTRL**+**X**, then **Y** to exit and save.

Restart the OpenSSH server with:

```
$ sudo systemctl restart sshd
```

06 Pair the arm with CDP

Open the Deploy Configuration tab for myCobotWiiRemoteController. Under Networks, press the Select button for 'WiFi'. The 'Devices – WiFi' table below should start showing any devices available to pair with CDP Studio. Click the Username field for your myCobot (based on its IP address) and enter 'er', then click the Pair button next to it. You will be prompted to enter the password – the default is Elephant.

Under Applications, for Device select your myCobot IP address or name, then change the Toolkit to ARMv8 64-bit (Debian 11). When you run the myCobotWiiRemote project, it will then be deployed over the wireless network to the robot arm.

07 Pair the Wii controllers

You'll need to pair the Wii Remote and Nunchuk controllers via Bluetooth with the robot arm's Raspberry Pi. First, open the battery compartment of the Wii Remote and press the small red Sync button to start pairing; the Remote's blue player LED should start blinking.

Now, from the robot arm's Ubuntu MATE desktop (viewed on a monitor or remotely via
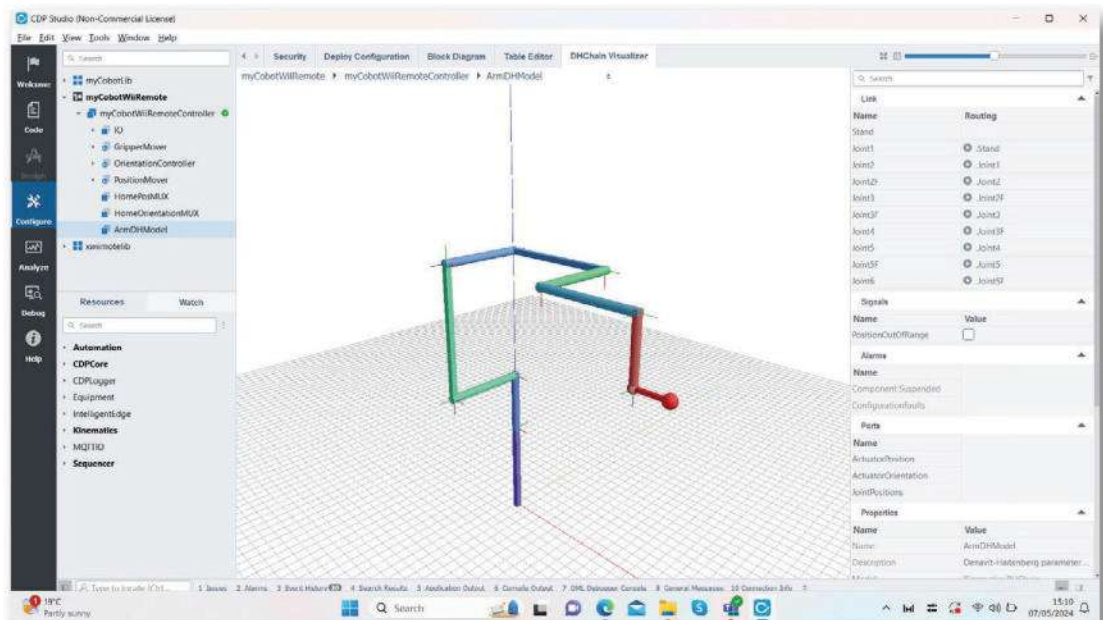
▶ You can view the arm's movements on-screen in the ArmDHModel DHChain Visualizer

VNC), open the Menu (top left) then search for and open Bluetooth Manager. Click Search and you should see the Wii Remote, possibly as 'Nintendo RVL–CNT–01'; select it and choose Pair. The Remote's LED should stop blinking.

You can see how the project's components connect to each other in the Block Diagram view

08 Run the project

Right–click myCobotWiiRemote in the left panel of CDP Studio and select Run & Connect. If you have a myCobot arm connected, it should follow the movements you make with the Wii controllers; if not, select ArmDHModel in the project's left panel to view a 3D representation of the arm with its six joints. The gripper status is indicated by a green (closed) or grey (open) dot.

By default, the arm's servos are released. Press the Remote's 2 button to engage them. Now hold the A button and tilt the Remote to tilt the arm's endpoint accordingly. Note that if you move outside the arm's range, the Remote will rumble. Pressing the Home button returns the arm to a preset home position.

The Nunchuk's joystick can be used to move the arm's endpoint forward, back, left, and right. When holding the Z button, you can also tilt the Nunchuk to move the endpoint up and down. Pressing the C button toggles the gripper status between open and closed.

To disengage the servos again, press the Remote's 1 button, but be ready to catch the limp arm as it falls!

09 Kinematics

This project makes used of CDP Studio's Kinematics framework, in the form of the DHChain component. The basic concept of kinematics is that if you input joint angles for a robot arm, or chain of links, the framework can calculate the end position in 3D space. This is used to control the arm's joint movements to reach a certain x/y/z position, such as for the home position and the those reached via the Wii and Nunchuk movements.

Kinematics has many uses in the field of engineering, helping to calculate positions and velocities of moving parts such as those in an industrial robotic arm, or a bionic limb or exoskeleton. An example real–world case is the use of CDP Studio and kinematics is for controlling deck cranes on ships.
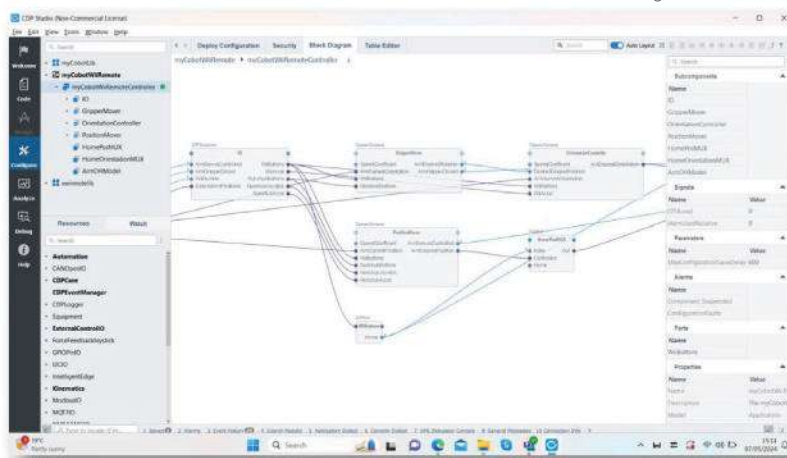
10 Exploring the project

You can click the Block Diagram tab to see how the project's block–based components have been put together. These include an IO block to interpret the controller actions, along with blocks for the desired arm position, gripper orientation, and gripper open/closed status. A couple of mux blocks enable the arm's home position and orientation to be triggered. The ArmDHModel block calculates the arm joint angles required for a required position, and also triggers the Wii Remote rumble if a position is out of range.

To take the project even further, you could add some extra actions for the unused controller buttons – by opening the **xwiimoteIO.cpp** code file and adding events and return codes for them there, then creating actions for the latter in the main application. 🎮

Top Tip 👍

No arm?

You can still run the project without a robot arm, by deploying it to a Raspberry Pi (or other Linux PC). Just make sure the ARM toolkit used matches the 32-bit or 64-bit architecture of its OS.

Using the Nunchuk, you can manoeuvre the arm's Adaptive Gripper and close it to pick up an object

Rescue your backups

Recover data from ageing floppy disks,
CDs and Zip disks before it's too late!

MAKER

K.G. Orphanides

K.G. still has all their
childhood floppy
disks, unfortunately
including the one
in the photo used
to illustrate an
unrecoverably
damaged floppy.

@owlbear

▶ **Beware:** Circular
scratch patterns on
a floppy disk can
indicate damage that
can in turn damage
your drive heads

This month, we're going to rescue hard-to-read media, from forensic disk imaging tools that attempt to recover data from your drive, bit-by-bit, to the important manual work of cleaning your disks and drives to restore deterioration and prevent further damage.

We'll take a close look at recovering data from CDs and floppy disks, helpful physical and software tools, and even help you get equipped to recover data from Iomega's Zip disk superfloppy formats. Please note that none of the techniques detailed here are a substitute for professional disk recovery, and that attempting to read damaged floppy disk media can harm both the disk and your disk drive.

01 Introducing ddrescue

In issue 140 (**magpi.cc/140**), we looked at using the Linux `dd` command to make byte-for-byte images of (non-copy-protected) floppy disks. You can also use it to create an iso file copy of a CD (once again, assuming it doesn't have copy protection). But if your storage media are too damaged to be fully read, then your image files will be unusable.

The GNU `ddrescue` data recovery tool is an entirely separate tool to `dd`, despite the similar name. It's designed to retry and hopefully rescue data from damaged areas of your medium. You can have it create a log (mapfile) that allows you to to interrupt and seamlessly resume it, and use tools to visualise the damage to your disk. We'll also use the `ddrescueview` mapfile viewer. Like `dd`, `ddrescue` only works on disk formats that your operating system can mount, which is generally limited to IBM PC formatted disks. Open a terminal and type:

```
$ sudo apt install gddrescue ddrescueview
```

02 Using ddrescue

`ddrescue` should not be used on a read-write mounted drive – we'll directly address the device name without mounting it. If your device is auto-mounted, unmount it before you begin the rescue process. We'll provide an example.

Insert your disk, and list your block devices with this command:

```
$ lsblk
```

Our floppy drive shows up as **/dev/sda**, while CDs appear at **/dev/sr0**. The MOUNT POINTS column indicates whether or not it's mounted. If required, we can unmount it thus:

```
$ sudo umount /dev/sda
```

The basic `ddrescue` command (including a mapfile) is `ddrescue [device block] [filename to save] [mapfile name]`. **/dev/sda** requires root permissions on Raspberry Pi OS, so type:
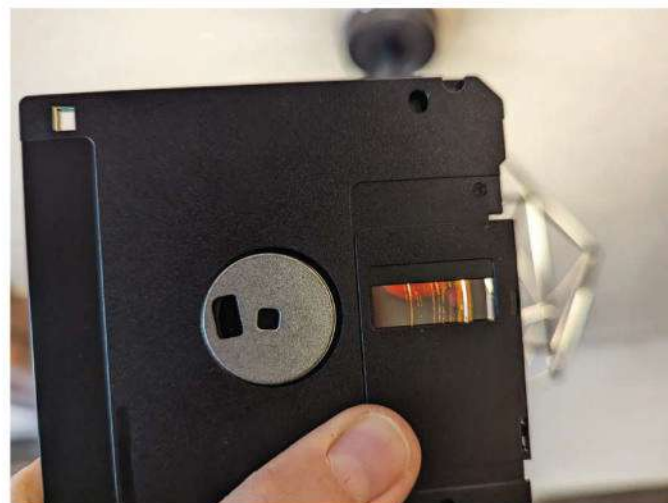
This 3D printed frame holds the disk open, allowing both sides to be accessed for cleaning with isopropyl alcohol
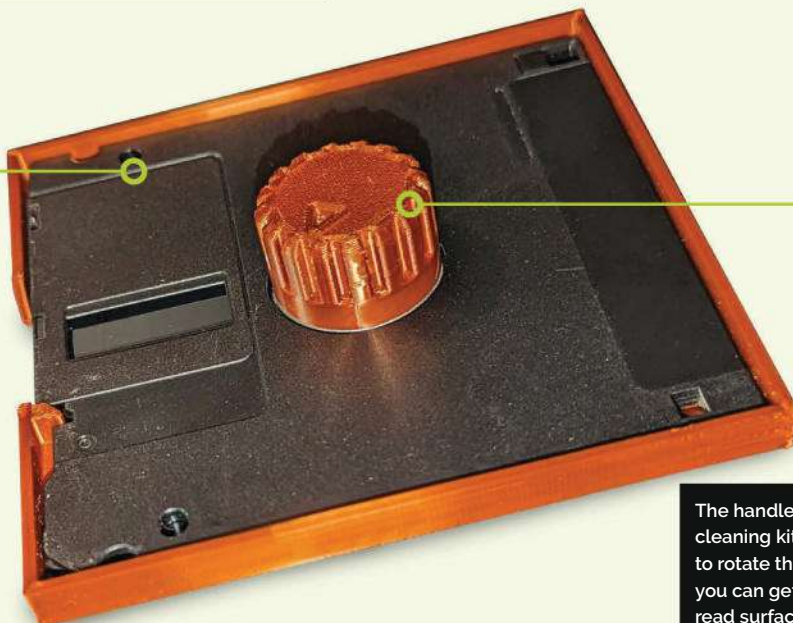
The handle part of the disk cleaning kit allows you to rotate the disk so that you can get to the entire read surface. It also helps you test that your disk can turn smoothly before putting it in a drive

Warning!
Solvents!

We discuss the use of isopropyl alcohol and cyclomethicone. Both are flammable solvents and eye irritants which should be kept out of the reach of children and animals to avoid accidental ingestion.

magpi.cc/solventsafe

```
$ sudo ddrescue /dev/sda floppy.img floppy1.log
```

Once the process is complete, you can give ownership of the files to yourself:

```
$ sudo chown YourUserName floppy.*
```

03 Check progress with ddrescueview

ddrescueview is a GUI tool that can open ddrescue mapfiles (previously known as logfiles – a term you'll still see reference to in ddrescue flags and conventions. You can start rescueview from your Raspberry Pi OS's Accessories menu, or from a terminal with the command:

```
$ ddrescueview floppy1.log
```

You can use it to watch ddrescue's progress and get an overview of a disk's rescue status. We also found it useful to create visual comparisons of rescue attempts using different drives.

You'll find the full ddrescueview manual at **magpi.cc/ddrescuedocs**

04 Merge two rescue images

ddrescue can also be used to merge two images. The command structure is:

```
ddrescue – m [mapfile to merge] [image to
merge] [merged image] [merged mapfile]
```

Repeat this for as many mapfile and image pairs as you have to merge, targeting the same merged image and mapfile each time.

This is particularly useful for recovering data using multiple drives. Different drives can read the media differently enough to each recover data that the other missed, allowing a more complete image to be reconstructed by combining their reads.

05 Cleaning a CD

Most CD read problems are the result of dirt or fingerprint smudges on the disc, which can be safely wiped off with a microfibre cloth. It is also safe to use water, alcohol and washing up liquid, individually or in combination, to clean

▲ Although it's possible to jury-rig adaptors for internal and even parallel Zip drives, USB drives are by far the most reliable way of accessing Zip disks on Raspberry Pi

▶ When working with floppy disks, it's a good idea to make them physically read-only. On most 3.5in disks, you just need to slide this switch to expose a square hole

CDs. Make sure to completely dry the media and ensure that it's streak-free. Immersing CDs in water is usually safe, but can, in rare cases, damage the top layer of the disc, so is best avoided unless strictly necessary.

If the damage is visible from the underside of the disc but appears to affect the top layer or even interior, this can't be cleaned or repaired, although we have managed to use `ddrescue` over a period of several days to recover data from a CD with the beginning stages of oxidation of the disc's data layer, also known as "disc rot" or "bit rot".

Top Tip 👍

RTM

The ddrescue manual can be found at **magpi.cc/ ddrescueman**

06 CD scratch removal

If your CD has significant scratches on the bottom, these can cause read errors. The best tool to correct these is a professional-grade CD resurfacing machine (**magpi.cc/ecopro2**), which removes and replaces the bottom layer of a damaged disk. In the UK, many branches of CEX have one and will resurface your discs for a small fee.

If you live somewhere where that's not an option, you can try to buff out scratches from your disk with Brasso metal polish, using a microfibre cloth and a circular motion. Ensure that you fully remove the polish from the disc with water after polishing. For a good look at the technique, see this video: **magpi.cc/polishdisc**

07 Print your own floppy cleaning toolkit

Floppy disks can degrade over time, shedding their magnetic surface, but more often, read errors are caused by dust or mould on the surface of the disk. This can be cleaned off with isopropyl alcohol, but you'll need a way to rotate the accessible surface of the floppy. Fortunately, the 3D printing community has you covered.

▶ On most 5.25in floppies, you'll need to cover the disk's write-protection notch with a sticker to make it read-only. These were provided with packs of disks, but any sturdy, opaque sticker or tape will do

Download both parts of Lemaru's 3.5in Floppy Disk Cleaning Tool at **magpi.cc/diskcleanertool** – they've also created cleaning tools for 3in disks (used by Amstrad in the UK) and 5.25in floppies, which, which you'll find on their Thingiverse account. They're simple prints, and Lemaru provides slicer configuration suggestions for Creality Ender 3 printers.

08 Clean your floppy disks

To clean your floppy, mount it face-down in the 3D-printed cradle with the metal disc cover open. The disk holder will keep this open. Line up the handle so that the square and rectangle slot into those on the disk's central hub. You'll now be able to rotate the internal disc for cleaning, giving you access to both sides of the disk. If necessary, blow dust away using a can of compressed air or an electronics cleaning blower. Clingier mould and dirt can be removed with 70% to 99% isopropyl alcohol on a microfibre cloth, on single-use alcohol wipes, or even a cotton bud if you're careful. We recommend higher volume alcohol, as it evaporates faster. Give the alcohol time to dry before inserting the disk into a drive.

09 Lubrication

A companion measure to cleaning your disk with isopropyl is lubrication. Apply a scant drop or two or of the silicone oil cyclomethicone and then rotate the disk to lubricate it. The Cyclomethicone evaporates particularly fast, but can temporarily smooth over damage sufficiently

to allow a floppy to be read, even in cases where a disk has visible scratches. Note that using solvents can accelerate the destruction of a disk's surface if it is already decaying and that trying to read a damaged disk can in turn damage your floppy drive's heads.
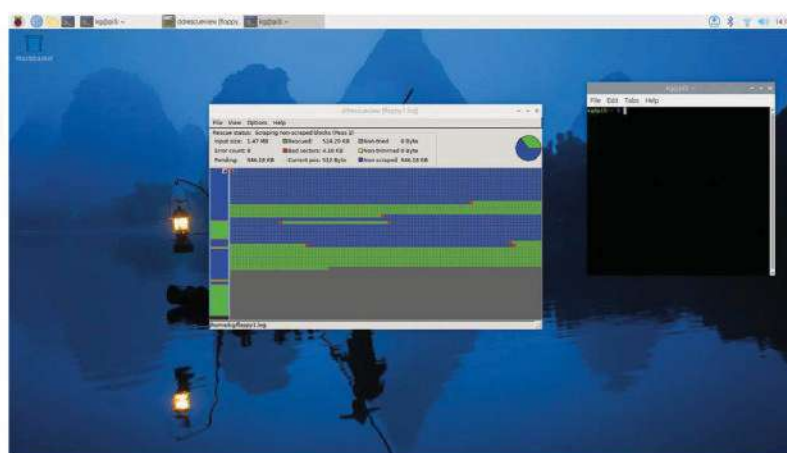
> **" Using solvents can accelerate the destruction of a disk's surface "**

▼ ddrescueview can open a map file created by ddrescue, allowing you visualise the status of your disk recovery

10 **Clean your floppy drive heads**

If you're going to be using dirty disks in your floppy drive, its drive heads can accumulate dirt. This can in turn cause read errors and even damage to disks that are inserted into them. The easiest solution to this is a floppy drive cleaning disk, but if you don't have one, you'll have to find second hand, refurbished or new old stock. While their accompanying bottles of isopropyl alcohol may have dried up, we've already got some of that.

You can also manually clean your floppy drive heads, either via the slot or, better, by disassembling it, which also allows you to lubricate its rails with white lithium grease, but this can damage the heads if not done with care and is labour-intensive. **M**

Reading Zip disks

In the late '90s, Iomega's Zip disks were briefly popular in academia, enterprise and music production as a high-capacity, rewritable storage medium. Zip drives would become the most widely used high-capacity 'superfloppy' disk format, until they were swept away by first writable and rewritable CDs and then solid-state storage.

If you need to retrieve data from a Zip disk, you should obtain a USB Zip drive, which will simply display the disk as a USB mass storage device on most modern computers.

The number in the names of each of the main three Zip generations – the Zip 100, Zip 250 and Zip 750 – indicates the capacity of its discs. The final generation of Zip drives, the Zip 750 released in 2002, had a capacity of 750MB. Each generation of drive can read both its own and the previous generation of Zip discs.

The Zip 250 can only write to Zip 100 disks slowly and the Zip 750 can't write to Zip 100 disks at all. Write speeds can, however, be disregarded for recovery purposes.

Zip 100 disks are the most common, but if you're looking to recover your own backups, you'll need to make sure you have a drive that is at least as high capacity as the disks you wish to read.

If you have a USB Zip drive, then you're in luck. Just plug it into Raspberry Pi – or indeed any modern PC

running Linux or Windows, or an older 32-bit Mac. If you hope to access a common parallel port Zip drive, you'll have more trouble. Although multi-purpose USB to parallel chips exist, the majority of USB parallel cables can only handle LPT printer connections. The discontinued StarTech ICUSB2321284 should do the job, but the newer ICUSB1284D25 does not, for example.

Similarly, internal ATAPI Zip drives don't appear correctly when connected to many modern IDE/SATA to USB adaptors, while older adaptors that only support IDE to USB are more likely to work. You'll also need a true ATAPI Zip drive, rather than earlier IDE Zip drives that didn't use the ATAPI protocol – this can be a bit hit and miss with Zip 100 models in particular. For a video guide to connecting internal Zip drives, see **magpi.cc/zipdriveusbide**

Zip drives from the '90s were prone to the 'Click of Death', which affected an estimated one in 200 drives at the time of a 1998 lawsuit. But the disks appear to be robust – we were able to read all of the 25+ year old Zip media we've tried, making Zip drives a useful way of getting large amounts of data onto old computers, once you've retrieved your precious file archives from them.

We were unable to obtain SCSI or FireWire models for testing, although, like USB models, the latter should work on any computer with FireWire support. Zip drives are not compatible with Jaz disks, a sibling product from Iomega.

Top Tip 👍

Format restrictions

If you need to copy disks for other computer systems, such as the Amiga or Atari ST, you'll want a flux copying device such as Greaseweazle
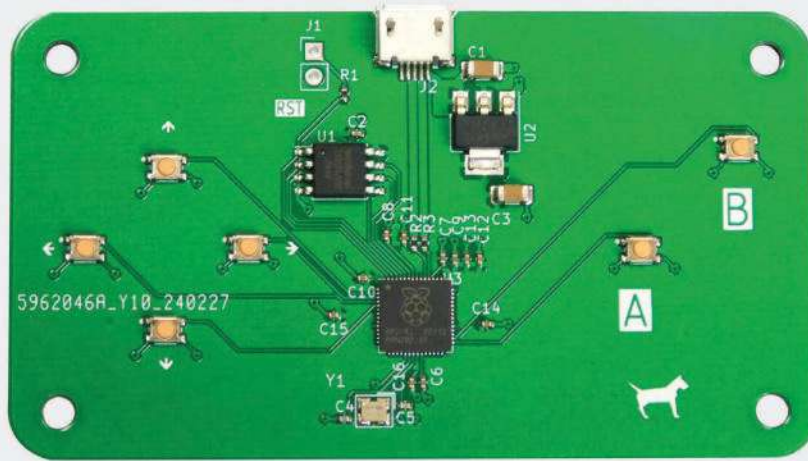
KiCad: making an RP2040 game controller

Let's explore adapting our RP2040 layout to make a USB game controller

Jo Hinchcliffe

Jo Hinchcliffe (AKA Concretedog) is a constant tinkerer and is passionate about all things DIY space. He loves designing and scratch-building both model and high power rockets, and releases the designs and components as open-source. He also has a shed full of lathes and milling machines and CNC kit!

Figure 1
The completed game controller PCB

n earlier articles in this series, we established that we have a reasonable working RP2040 layout, so now it's pretty trivial to create new RP2040 devices. In the last section we made an 'Urumbu'-style motor driver board, and in this section we are going to create a simple USB game controller (**Figure 1**).

It's largely the same process we undertook for the Urumbu project, but simpler. We started by making a copy of our Urumbu project and cleaned out the files in the new project copy that we wouldn't need. So, any particular files like the board edge geometry, the Gerbers, and CSV files can be deleted as we will replace them with ones generated for the new project. We also quickly deleted all the Urumbu parts we didn't need from the schematic. Note that we don't really need to delete items in the PCB Editor, as when we eventually pull in the updated netlist and bill of materials, we can automatically delete unreferenced footprints, and the new footprints will be brought in.

We want to add six tactile buttons to our RP2040: four in a D-pad arrangement and two as A- and B-style buttons. We want these buttons to be momentary press buttons and 'push to make'. We then plan to use these buttons to connect one side to a GPIO and the other side of the button to ground.

Scouring the JLCPCB parts library, we came across the C221902 button. This part looked a nice size, so we took a look at the EasyEDA schematic and footprint. It has four pins and, reading the schematic, we could see that if we connected pin 2 to a GPIO and then connected all the other pins to ground, it would work as we wanted. Additionally, with the four SMD pads, it should mechanically be pretty strong.

With our choice made, we used the excellent Wokwi EasyEDA 2 KiCad website to convert the supplied footprint to a KiCad format: (**hsmag.cc/easyEDA2KiCad**). We then added it to our custom library. We covered this in earlier sections of this series, but it's pretty straightforward. You upload the EasyEDA JSON file, and it then downloads a KiCad PCB file with the footprint loaded into it.
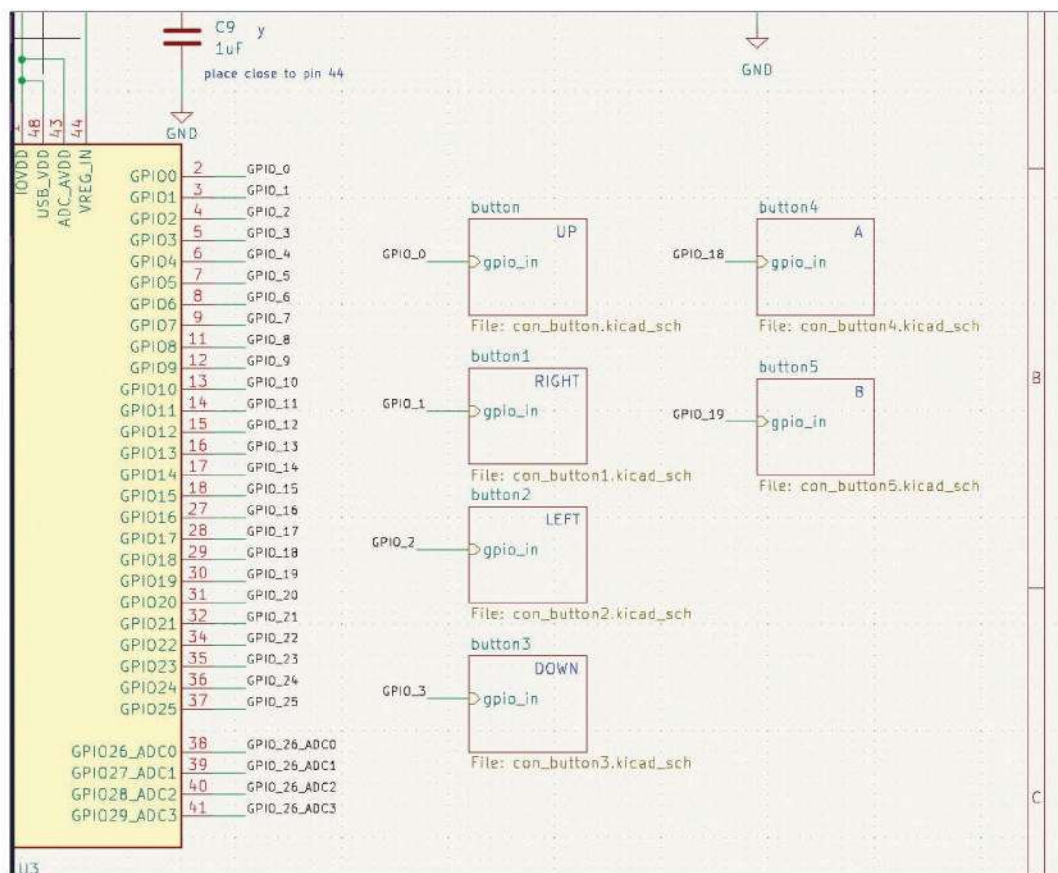
To add the buttons to our schematic, we created a custom 4-pin schematic symbol and inserted it into a hierarchical sheet. We wired the GPIO pin and the other pins to ground and then brought out the GPIO hierarchical pin. We then copied the hierarchical sheet to create six versions, one for each button, adjusting the label and the sheet name as we added each (**Figure 2**). Again, we've covered this in earlier sections.

Next, we assigned the new footprints to the schematic symbols and began to edit the PCB layout. We created a new board edge geometry SVG in Inkscape with some mount holes before carrying out the usual exporting of Gerbers, BOM, and positional files for JLCPCB services (**Figure 3**).
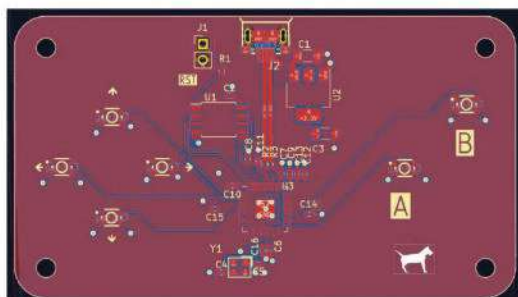
> **"** Additionally, with **the four SMD pads,** our PCB should mechanically be pretty strong **"**

Having ordered the boards, one final fun activity on the hardware side of this build was to export a STEP file from KiCad to model around in FreeCAD. To export a basic STEP file from the KiCad PCB Editor, select File > Export and then choose STEP as the output format. Note that we haven't added custom 3D models for all of our custom components, so obviously the STEP file isn't completely correct, but it serves as a good enough guide to model around in FreeCAD. ➜

HIGH PERFORMANCE

In this tutorial, we've looked at creating a gamepad that's easy to understand and extend. However, if you're looking to build a high-performance gamepad, then there are lots of things that you need to take into account. Part placement is obviously a large part of it, as you need to be able to press buttons consistently and accurately.

However, another part is the software. Our CircuitPython code could be improved, but ultimately, if you're looking for high performance, CircuitPython isn't the right choice. Fortunately, there is another option.

GP2040-CE is a firmware for RP2040-based devices. You can configure it with details of what hardware is connected where. It understands more than just buttons, so you can add analogue inputs as well.

There's documentation on the project website: **hsmag.cc/GP2040-CE**.
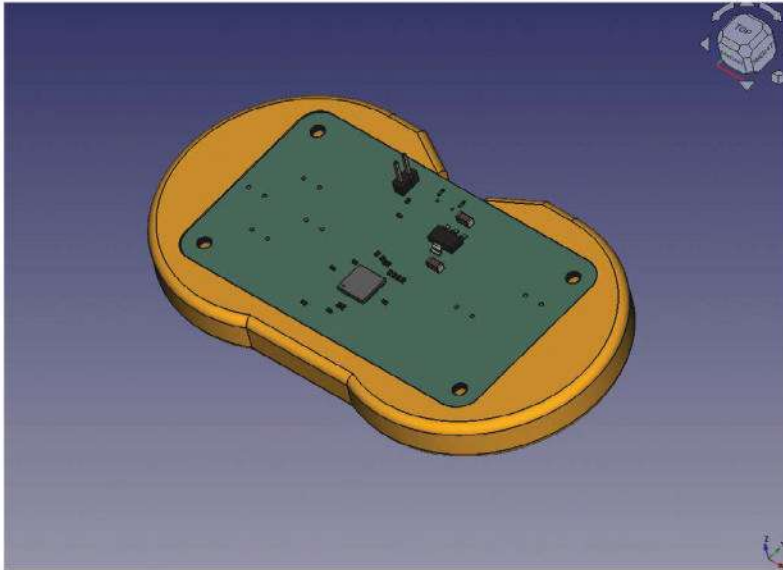
In the free-to-download book *FreeCAD For Makers*, we explored using FreeCAD and the KiCad StepUp workbench that allows you to create and position custom 3D component models for use in KiCad. We also explored all the skills needed to create all kinds of models. With the knowledge you gain from this book, you could certainly make a controller enclosure like the one we quickly modelled (**Figure 4**).

THE SOFTER SIDE

Now we have created our board, it's time to write some code for it. We could write our code in C using the Pico SDK. We could also use the Pico build of MicroPython or CircuitPython. However, since we've created a new board, let's create a firmware tailored specifically for it – we'll create a custom build of CircuitPython. This allows us to do a couple of things. Firstly, it lets us name the specific pins, so rather than using, say, GPIO0, we can use BTN_A. Secondly, it lets us select which modules we want to include. In our case, we'll add Adafruit HID, which enables us to use our game controller as an input device.

The general process for creating a build of CircuitPython is given in the documentation at **hsmag.cc/BuildCP**. We won't go through it in detail, so follow that guide to set up your environment.

Once you have everything set up, you need to create this board. In the directory **circuitpython/ports/raspberrypi/boards**, copy the Raspberry Pi Pico directory into a new one that's named appropriately for the gamepad. We've called ours **hackspace_gamepad**.
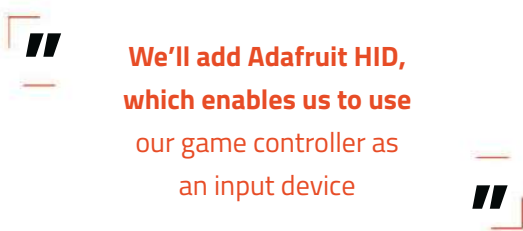
There are two files that we need to adjust to take into account our board. Firstly, there's **pins.c**, which should have the following:

```
#include "shared-bindings/board/__init__.h"

STATIC const mp_rom_map_elem_t board_module_
globals_table[] = {
    CIRCUITPYTHON_BOARD_DICT_STANDARD_ITEMS

    { MP_ROM_QSTR(MP_QSTR_UP), MP_ROM_PTR(&pin_
GPIO0) },
    { MP_ROM_QSTR(MP_QSTR_RIGHT), MP_ROM_PTR(&pin_
GPIO1) },
    { MP_ROM_QSTR(MP_QSTR_LEFT), MP_ROM_PTR(&pin_
GPIO2) },
    { MP_ROM_QSTR(MP_QSTR_DOWN), MP_ROM_PTR(&pin_
GPIO3) },
    { MP_ROM_QSTR(MP_QSTR_BTN_A), MP_ROM_PTR(&pin_
GPIO18) },
    { MP_ROM_QSTR(MP_QSTR_BTN_B), MP_ROM_PTR(&pin_
GPIO19) }
};
MP_DEFINE_CONST_DICT(board_module_globals, board_
module_globals_table);
```

In this, we're adding items to the board module. Specifically, one for each button.

> **We'll add Adafruit HID, which enables us to use** our game controller as an input device

OTHER **GAMEPADS**

This example should get you started in the world of game controllers, and there are loads that you can look at for inspiration:

- The Arduino Esplora is now retired, but was one of the first hackable game controllers on the market: **hsmag.cc/ArduinoEsplora**
- There's an online community at PCBWay's shared projects site that includes many game controllers, including: **hsmag.cc/PicoGamepad**
- Gamepads come in many shapes. They're usually designed around ergonomics, but you can get a little creative. For example, this maker has built a bat-shaped controller: **hsmag.cc/BatController**

Next, we need to edit **mpconfigboard.mk** to be the following:

```
USB_VID = 0x1209
USB_PID = 0xB182
USB_PRODUCT = "HackSpace gamepad"
USB_MANUFACTURER = "HackSpace magazine"

CHIP_VARIANT = RP2040
CHIP_FAMILY = rp2

EXTERNAL_FLASH_DEVICES = "W25Q128JVxQ"

CIRCUITPY__EVE = 1


FROZEN_MPY_DIRS += $(TOP)/frozen/Adafruit_
CircuitPython_HID
```

In this file, we define the type of flash chip we have and also add any 'frozen' modules we want. Frozen modules can be anything that we want to be included on the build by default (other than the core modules that are automatically included). Frozen modules have to be in the **circuitpython/frozen** directory, but you should find that the Adafruit_ CircuitPython_HID module is already there.

You can now create your build by going to **circuitpython/ports/raspberrypi** and running:

```
make BOARD=hackspace_gamepad
```

This will compile your code, and you should end up with a **build-hackspace_gamepad** directory. In there, you'll find a **firmware.uf2** file that you can load onto your gamepad just as you would any other UF2 file.

Obviously this isn't complete firmware as it's only the programming language. We now need to write a program to get everything working. Fortunately, we have all the modules we need baked in, so there's no need for anything there. We've drawn inspiration from the CircuitPython example code here: →

```
import time
import board
import digitalio
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import
KeyboardLayoutUS
from adafruit_hid.keycode import Keycode

# A simple neat keyboard demo in CircuitPython

# The pins we'll use, each will have an internal
pullup
keypress_pins = [board.UP, board.DOWN, board.LEFT,
board.RIGHT, board.BTN_A, board.BTN_B]
```

LEAD-FREE

It's often cheaper to get boards made using leaded solder. However, this might be a false economy. Leaded solder is harmful to both your health and the health of our planet. In the case of a games controller – something that you're going to hold in your hand time and again – it's more important than usual to opt for lead-free solder. Even if only a tiny amount gets on your hands each time you use it, that will still add up over the course of the controller's life, and could have negative effects on your health.

```python
# Our array of key objects
key_pin_array = []
# The Keycode sent for each button, will be paired
with a control key
keys_pressed = [Keycode.UP_ARROW, Keycode.DOWN_
ARROW, Keycode.LEFT_ARROW, Keycode.RIGHT_ARROW,
Keycode.A, Keycode.B]

# The keyboard object!

time.sleep(1)  # Sleep for a bit to avoid a race
condition on some systems

keyboard = Keyboard(usb_hid.devices)
keyboard_layout = KeyboardLayoutUS(keyboard)

# Make all pin objects inputs with pullups
for pin in keypress_pins:
    key_pin = digitalio.DigitalInOut(pin)
    key_pin.direction = digitalio.Direction.INPUT
    key_pin.pull = digitalio.Pull.UP
    key_pin_array.append(key_pin)

print("Waiting for key pin...")

while True:
    # Check each pin
    for key_pin in key_pin_array:
        i = key_pin_array.index(key_pin)
        key = keys_pressed[i]
        if not key_pin.value:  # Is it grounded?
            print("Pin #%d is grounded." % i)
            # "Type" the Keycode or string
            keyboard.press(key)  # "Press"...
        else:
```

Right ◆
Our custom build of CircuitPython brings everything we need, including pin names and modules
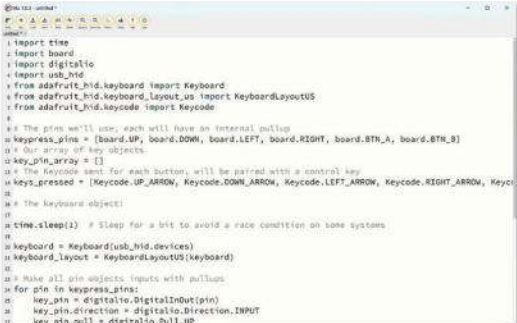
```python
            keyboard.release(key)
    time.sleep(0.01)
```

As you can see, we can use **board.UP**, **board.DOWN**, **board.LEFT**, **board.RIGHT**, **board.BTN_A**, and **board.BTN_B** in our code. This has a couple of advantages. Firstly, it is more intuitive for programmers. Secondly, if we created another version of the board with the buttons on different pins, the same code could still run on both.

This code is a bit lazy. For example, there's no debouncing on the buttons. In practice, we've found that this doesn't cause many problems, especially with the **time.sleep(0.01)** in there. This means it's not the most responsive controller, so if you're playing games where hundredths of a second matter, you probably want to use something different, including tuned debouncing, written in C. However, this controller isn't suitable for that type of game anyway. This is also fairly cavalier with the number of reports it sends (a report being a status update sent from keyboard to computer). This will send six of them every loop, which means several hundred a second. Again, this isn't great for performance. However, it works reliably and is easy to understand.

With this code loaded, you should be able to plug the controller into any computer and it will recognise it as a USB keyboard. Press one of the buttons and the computer should recognise that button press just as it would from any keyboard. With this, you can control any game that takes input from a computer.

Creating a custom version of CircuitPython isn't essential when you build a new board; however, once you've been through the process once, it's easy, and makes life a little bit nicer, especially if you're distributing the board to other people. ☐

Summer projects

Summer is here, so what better time to start thinking about which Raspberry Pi projects can help make the sunnier months even more pleasurable?

By David Crookes

Whether you're going away or planning a staycation over the summer, one thing's for sure: these are the best months of the year for getting out and about. The better weather means you can enjoy your garden, consider having an adventure or spend some time sensibly soaking up the rays by the coast. But you don't necessarily have to abandon your Raspberry Pi. As we're about to see, our favourite computer can be used to create a host of useful projects that can give you more time to have a great time. Happy days await.

Garden projects

Spend less time maintaining your garden and more time enjoying all it has to offer – weather permitting, of course

Rain detector
magpi.cc/summerrain

Summer isn't always a guarantee of sunshine – you just know the moment you get the barbecue out, decide to mow the lawn or put the washing on the line, rain will begin to fall. With this simple rain detector powered by a Raspberry Pi Zero W you can receive an alert straight to your mobile phone whenever there's about to be a sudden downpour, giving you time to take action. It's highly effective, hyper-local and rather inexpensive too. And since it doesn't require any soldering, it's a perfect project to get you started.

▲ You can make use of any good-sized watertight container you may have lying around

◀ The rain water sensor modules can be picked up very cheaply online

PiMowBot
magpi.cc/summermow

If you don't like mowing the lawn, there are generally three alternatives: you let the grass grow, employ a gardener or invest in a robot mower. If the final option appeals, then you may find yourself shelling out at least £450 – unless, that is, you decide to build your own based around a Raspberry Pi Zero. The PiMowBot, which also includes a Raspberry Pi Camera Module and a solar panel, has evolved over recent years and, while the management software remains a work in progress, the build shouldn't cost you more than £300. It has also been created to last with no real wearing parts, battery aside, so you'll be able to relax with a cool drink while it does its thing for many years to come.

▲ Most of the parts including the PiMowBot's housing and chassis are 3D printed

▶ All of the electronic parts are neatly tucked away inside the mower's housing

Raspberry Pi Plant Watering
magpi.cc/summerwater

The sunny days can leave your plants very thirsty so it's important to keep them regularly watered. With Christopher Barnatt's amazing system, built around a Raspberry Pi Zero computer, you don't need to set reminders or make arrangements if you're going away. You can just allow a moisture sensor to check that a plant is sufficiently hydrated every 30 minutes then let water automatically flow if the soil is too dry. Water flow is controlled by a solenoid valve and, as a fun bonus, the build includes a camera so that you can record a time-lapse movie of the growing plants.

▲ It looks like an elaborate system but Christopher talks you through the whole setup process

◀ A resistive moisture sensor costing just £4 is connected to Raspberry Pi

Bird Feeder Monitor
magpi.cc/summerbird

During the Summer, lots of wildlife may visit your garden. Some animals are likely to be unwanted – foxes and badgers can tear gardens apart so you may want to deter them (**magpi.cc/summerdeter**) but most, such as birds, are very much welcome. Stephen B Kirby's Bird Feeder Monitor detects our feathered friends as they perch ready for some food before taking photographs of them in action. The images are then stored along with information about the local humidity, cloud cover, temperature, wind speed, gust and direction, allowing you to work out when different species of bird are most likely to visit.

◀ A CAP1188 turnkey capacitive touch sensor is located within the weatherproof box

▼ Stephen has managed to use the project to capture some stunning natural images

Adventure projects

Raspberry Pi can be the perfect companion whether you're hiking, camping or simply looking for lots of fun outdoors

▲ The Raspberry Pi is housed inside the bag within a metal case to dissipate heat

▶ Thanks to the accessible connector on the side, it's easier to turn on and power the bag

Raspberry Pi Backpack
nicholashacault.com

When you're out and about on your travels this Summer, you may yearn to have your Raspberry Pi close to hand. If that's the case then this project literally has your back, allowing you to take your favourite computer on any adventure you desire. Created by Nicholas Hacault, it incorporates a seven-inch screen into a backpack that's linked to a Raspberry Pi. Operated using a wireless keyboard and mouse, and juiced via a 30,000 mAh portable power bank, it could even include a walking animation connected to an accelerometer. And since Raspberry Pi devices are small, there's plenty of room for all of your other stuff too.

Recovery Kit Version 2
magpi.cc/summerrecovery

Originally created five years ago by Jay Doscher, this tough cyberdeck was designed to withstand pretty much anything that could be thrown at it. Incorporating a seven-inch Raspberry Pi touchscreen, thereby removing the need for a mouse, it effectively placed a Raspberry Pi 4 inside a small, air and watertight Pelican case and allowed it to be powered using 5V. More recently, Jay has revisited the project using Raspberry Pi 5, a large battery and new Drop/OLKB Planck v7 keyboard. He's also made it easier to build, and you could easily adapt this one to suit your needs. It sure beats having to worry about damaging an expensive laptop.

▲ The build makes use of an ortholinear keyboard. These have the keys aligned to a strict grid

▶ The screen sits in its own compartment which can be pulled from the Pelican case for access.

ML Clip Cam
magpi.cc/summerclipcam

This project isn't your usual Raspberry Pi HAT. Instead, it's a cool camera device that works with a hat of the more traditional variety – you know, the ones you wear on your head. Created by Jacob David C Cunningham, the lightweight camera clips on to the peak of a cap, allowing you to make easy point-of-view recordings of your adventures. It would also be possible to incorporate machine learning into this project by getting it to identify particular objects on your travels and make a log of them (the ML Clip Cam is able to identify RC planes in real time). With a bit of time, you may be able to create a case for the camera, particularly if you're fashion conscious or want to protect the Raspberry Pi Zero 2 W from the elements.
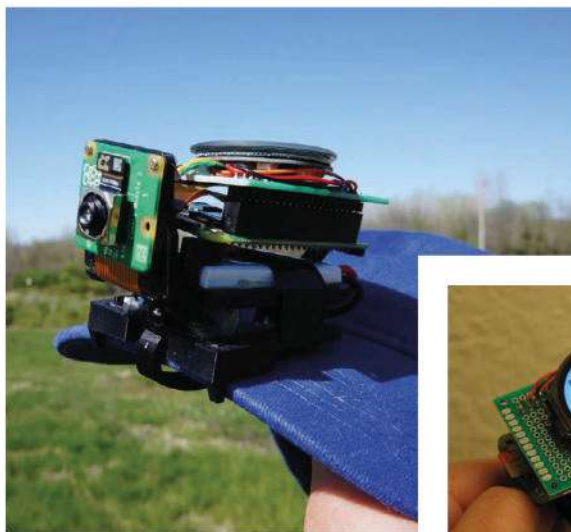
◀ It still looks like a prototype but get your thinking cap on – you may be able to work out a way to make it look pretty

▼ An ESP32-S3 1.28-inch touchscreen is incorporated into the project

▲ The passphrases are displayed on Pick-Clock-Green LED-digit electronic clocks designed for Raspberry Pi Pico

▶ The aim of the game is to fix a 'broken' TARDIS by entering the correct combination of passwords in the correct sequence
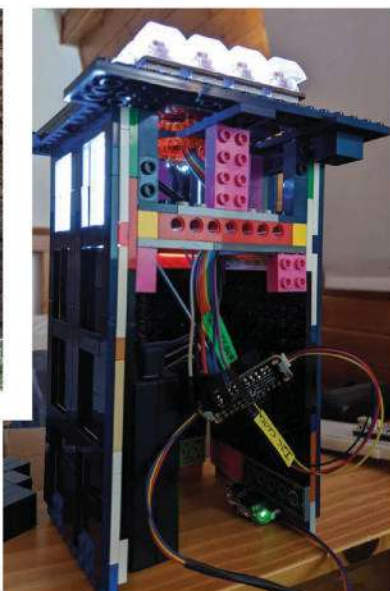
TARDIS Treasure Hunt
magpi.cc/summertardis

You can add another dimension to any adventure by incorporating a game and, if fun's what you're after, Roberto Tyley's project can certainly provide it. He hid two screens in a local park, each of which could be used to generate one half of a passphrase. When the time-limited passphrases were combined, they could be entered into a TARDIS built out of LEGO – this would cause the windows to gradually light until they were all lit, thereby completing the game. As well as requiring three teams (one situated at each of the passphrase generators and another at the TARDIS, all communicating by walkie-talkies), the project also needed a trio of precision real time clocks and some nifty coding so that the system could work out if the code combos issued at any one time were correct. The Time Lords would be proud.

Wish you were here!
Many of us like to share our adventures with others and Raspberry Pi can help you. As well as taking quality snaps using your own, cool, DIY camera, such as the retro-styled RUHAcam (**magpi.cc/ruhacam**), you could produce your own DispatchPi (**magpi.cc/dispatchpi**) – a digital e-ink photo frame that allows you to instantly share black-and-white images using photos emailed to a Google Mail inbox. You can also capture any action at 90fps with the piOneer action camera (**magpi.cc/pi0neer**) which is a great low-cost alternative to a GoPro.

Seaside projects

Who doesn't love to be by the seaside? Use Raspberry Pi to help you make the most of your time on the beach

❝ Nobody wants to arrive at the beach only to discover that it's not there! ❞

Tide Tracker
magpi.cc/summertide

Nobody wants to arrive at the beach only to discover that it's not there! Likewise, you don't want to aim for a dip in the sea and find that it's actually far away. In either case, it's so much more convenient to get a handle on the times for high and low tides and that can be achieved using a tidal forecast API from the Admiralty in conjunction with Raspberry Pi. Add a screen such as an Inky pHAT display, find your tidal station, tinker with some code and you're effectively good to go. The display will tell you the current situation and indicate the time it's due to change.

◀ You will need to attach a GPIO header to your Raspberry Pi

▼ The tide tracker works with a tidal stations of your choice and shows the rise and fall of the water

Pico Held
magpi.cc/summerheld

You can take your pick from scores of maker-led handheld games consoles but we do like the Pico Held, primarily because it's open source, based around our favourite flexible microcontroller board and is capable of playing titles developed for the Sega Mega Drive. Its potential low cost also means you're less likely to worry about sand infiltrating the case or accidentally dropping it in the sea, allowing you to simply kick back on the beach and concentrate purely on the games. We like that the 3.2-inch display is large enough to see (the games should be bright enough even if you're wearing sunglasses, too). The device's handheld library will also allow you to create your own classic pixel art games if you fancy getting creative.

▲ The device has three buttons and an analog control stick for interacting with your games

◀ The Pico Held is a smart-looking device that may well attract envious looks on the beach

Drinks machine
magpi.cc/summerdrinks

Being by the seaside in hot weather can make you thirsty, so it's a good idea to take a refreshing drink with you. Rather than put some concentrated juice in a flask and add water from the tap, however, you could create this fun drinks machine. Originally designed to sit on a desk and pour a fresh glass of squash when you're working, it lets you press a button to activate a relay and switch on a small submersible pump. The drink will be created and its name will be displayed on the screen, ready for you to take on your travels. It can even be set up to dispense drinks at set times – great if you're going in and out of the garden and need some timely top-ups.

▲ The drinks machine's case was designed in Fusion 360 and 3D-printed using a red filament

▶ The project uses Raspberry Pi, allowing you to SSH into the device and select a drink from afar

▲ The flight tracker shows the airports tracked planes are travelling between, as well as the airline logo

▲ When there are no planes, the current temperature is displayed every 10 minutes and forecasts are updated every hour

Flight tracker with weather
magpi.cc/summerflight

Want to enviously cast your eyes skywards at people traveling abroad for some time by the sea? Then this flight tracker is for you. It draws information from FlightRadarAPI and compares the lat/long data of planes with your precise location. Whenever a plane passes overhead, it'll display the flight route, airline, distance and direction on a 64x32 RBG matrix panel. If the skies are empty, it'll show the current time, date, temperature and three-day weather forecast – perfect if you're planning a trip of your own. More recently, creator Adam Paulson has added an auto-dimming option as well as a new collection of airline logos.

Look after your home

If you're going away this summer, you'll still need to keep a check on your home. Those with pets who can't travel may want to try Rebecca Peck's Raspberry Pi-powered pet feeders (**magpi.cc/summerpets**) and those worried about burglars should consider making an intruder alarm (**magpi.cc/summeralarm**). You could communicate with visitors while you're out and about using a smart doorbell and video intercom system (**magpi.cc/summerdoorbell**). And you can help prevent people nicking parcels from your doorstep using the package thief deterrent (**magpi.cc/summerpackage**). M

Crow**Vi**

▶ Elecrow ▶ **magpi.cc/crowvi** ▶ From £92 / $115

Is this the ultimate portable monitor for Raspberry Pi?
Rob Zwetsloot takes a look at this interesting display

▶ The cover can be removed if you don't plan to use it

Verdict

A great monitor in its own right that performs well out of the house and well enough in direct sunlight

8/10

Portable monitors are a slightly odd beast – you're expecting a display as good as the one at your PC desk, yet it also needs to be compact and easy to set up at your destination. However you only have to look at adjacent products to see it done well – say, such as with an iPad and its smart cover.

Elecrow, maker of the CrowVision screens we've reviewed in the past, has taken a little inspiration from this to create a small yet functional monitor line that comes with its own cover and stand. It comes in a range of sizes and resolutions, with and without touch control. We have one of the 1080p 13.3-inch non-touch versions to review, however there's also a 4K version and a 15.6-inch alternative, all with the cover/stand included.

Plug and play

All versions of the CrowVi are able to be used out of the box with no extra set up – touchscreen models require you to use the USB-C port (not the power one) so the connected Raspberry Pi (or PC) can detect the inputs – but all models just need power and a HDMI connection.

There's no battery installed in the CrowVi so you'll have to provide external power, portably with a mobile battery or powered by the device you're connecting it to at home if you don't have a spare plug. To cut down on cables and power sources, CrowVi suggests daisy chaining power by using the USB-C data port on the display to power a Raspberry Pi while out and

If your smartphone supports display out, it's a great way to have a portable video player

about. You won't be able to get Raspberry Pi 5 up to full power this way, however it will run well enough.

As an aside, we understand the touchscreen is much like a standard touch display and is not sensitive enough for drawing, but does have the functions you'd otherwise expect from a smartphone or tablet.

Out and about

With unseasonable warmth during the review period, we were able to take the screen out and test it in the most extreme of conditions: direct sunlight. IPS displays do tend to have better visibility out in the sun and this screen performed fine, just as you'd expect any display to work. Make sure to crank up the brightness but find shade if you can.

As for power draw, a beefy power bank was able to keep everything running fine for a couple hours. Brightness levels and speaker use will affect this, so using external speakers or headphones will also change power draw.

The colours and responsiveness of the screen are great too. The image is sharp, and it was as good as any monitor we use at home. The cover is very stiff, and the magnets attaching it very strong, to make sure the display is held in place when propped up – it may be a touch too stiff at the bend and we weren't always sure it was in standing position. Over time, though, it has loosened up enough.

The sturdy construction feels and looks great too – and as we're home bodies we're currently looking into how we can integrate it as a third screen in our set up, which is pretty high praise we think. **M**

> **"** The colours and responsive of the screen is great too. Colours are sharp and it was as good as any monitor we use at home **"**

▲ It works great with a Raspberry Pi 5, and there's even a kit that comes with Raspberry Pi 400

10 amazing:

Gaming accessories

Enjoy your Raspberry Pi games more with these excellent upgrades

We meet a lot of people who use Raspberry Pi for gaming. Whether on the go or in their living room, there's so many ways to play with Raspberry Pi and even Pico. We've put together a list of some items that are sure to improve your experience.

▲ Adafruit Joy Bonnet

Miniature console

Turn your Raspberry Pi Zero into the tiniest plug-and-play console ever – all you need is a screen to plug it into!

magpi.cc/joybonnet | £12 / $15

▲ Adafruit Arcade Bonnet

Stripped down arcade

Create a small yet powerful arcade machine out of Raspberry Pi Zero with this minimalist add-on for JST connectors

magpi.cc/arcbonnet | £12 / $15

◄ Pico Dongle Lite

Custom input

If you've made custom input devices for games with a Pico, this is another great way to interface that Pico with the device you're gaming on

magpi.cc/picodongle | £2 / $3

▼ Game 5Pi Retro Gaming Case

Classic enclosure

Upgraded to the latest Raspberry Pi 5 but still want to rock a retro case? Game 5Pi allows you to sneakily camouflage your Raspberry Pi 5 as something resembling a NES. It also comes with a heat sink fan kit.

magpi.cc/game5pi | £17 / $21

▲ Classic wired joypad

Nostalgic control

Sometimes the best way to play old games is with a controller you remember. No fancy analogue sticks, just eight buttons and a D-pad

magpi.cc/classicpad | £14 / $17

▼ PiStation Case + LCD

Portable console

A throwback to the '90s, this console-plus-screen combo is definitely more powerful than the machine it's emulating, but that just makes it cooler

magpi.cc/pistation | £81 / $101

◄ 8BitDo Wireless Adapter 2

Wireless gaming

Connect all manner of Bluetooth controllers to your older Raspberry Pi (or any other system!) thanks to this powerful dongle – it also supports gyro controls

magpi.cc/8bitdongle | £19 / $24

▼ PiBoy XRS

Handheld power

This case turns your Raspberry Pi into a handheld console that feels like a top quality official product. Very little assembly required too!

magpi.cc/coolercover | £3 / $4

▼ 8BitDo Pro 2 controller

Wireless gaming

Our favourite retro-styled controller got even better with the Pro 2. Not only will it sync with a Raspberry Pi but also your Nintendo Switch too

magpi.cc/pro2 | £42 / $52

▼ Picade X HAT

Arcade heart

If you're planning to build an arcade based on Raspberry Pi, the Picade X HAT is the ultimate way to connect it all together

magpi.cc/picadex | £16 / $20

Learn networking
with Raspberry Pi

Resources to help you understand computer networking. By **Phil King**

An Introduction to Computer Networking for Teachers

CREATOR

Raspberry Pi Foundation

Price:
Free

**magpi.cc/
rpfnetwork**

Computer networking is a vital part of modern life, enabling us to browse the web and communicate with people all over the globe, but do most of us really know how it works behind the scenes?

Aimed at educators, specifically GCSE/high-school computer science teachers, this Raspberry Pi Foundation course – now available via edX – gives a good overview of the topic of networking. Over the three-week duration (although you can complete it at your own pace), you will learn about the different types of computer networks, and their pros and cons.

Key topics include the transmission of data, covering IP packets and networking protocols such as TCP and DHCP, along with cybersecurity.

You'll also gain a greater understanding of how the internet works, including routing, DNS, and the World Wide Web. The course content comprises mainly illustrated text and a few videos, including some cool animations. M

Computer Networking: A Top-Down Approach

CREATORS

James F Kurose, Keith Ross

Price:
£67 / $80

**magpi.cc/
topdownnet**

This hefty 800-page guide is recommended reading for computer science students and often used to accompany courses. Like most textbooks, it can be a bit dry at times, but it's approachable enough for beginners, and offers an excellent foundation to understanding the key concepts in the field of computer networking.

Now in its eighth (global) edition, the book has been updated to reflect recent advances in networking, such as the importance of software-defined networking (SDN) and the rapid adoption of 4G/5G mobile networks.

Starting with an introductory overview of the internet and computer networks, its subsequent chapters cover the

principles of network applications, transport-layer services, the network layer (data and control planes), link layer/LANs, wireless/mobile networks, and cybersecurity. Use it as a reference book or read it all to obtain comprehensive knowledge of networking. M

Useful websites

Check out the wealth of networking resources online

DEBIAN REFERENCE
Since Raspberry Pi OS is based on Debian Linux, the networking section of this online reference manual is ideal for learning a host of useful commands.
▶ **magpi.cc/ debianrefnetwork**

LINUX KERNEL
The networking section of the Linux Kernel documentation is packed with useful info on all aspects of Linux networking, from AF_XDP to XFRM.
▶ **magpi.cc/ kernelnetworking**

RASPBERRY PI FORUMS
If you're having networking issues with your Raspberry Pi, the 'Networking and servers' section of the official forums is a good place to go for help.
▶ **magpi.cc/ etworkingforum**

Mastering Python Networking

CREATOR

Eric Chou

Price:
£38 / $50

magpi.cc/ pythonnet4e
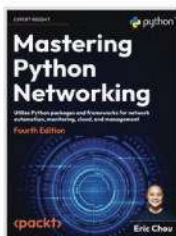
This 594-page book is aimed at IT professionals and operations engineers, so you'll need at least a basic knowledge of Python programming and networking to get the most out of it. The updated fourth edition features extra chapters on Docker containers and Python 3 Async IO for network engineers. All of the other chapters are updated with the latest libraries with working examples, too.

Starting off with Python fundamentals, the book then explores its applications in legacy and API-enabled network devices. It demonstrates making use of Python packages for network automation, monitoring, management, and security. The content progresses to cover AWS and Azure cloud networking. You'll also learn Git for code management, GitLab for continuous integration, and Python-based testing tools for network verification. **M**

Online courses

Enrol in a computer networking web course today

INTRODUCTION TO OPEN SOURCE NETWORKING TECHNOLOGIES
This edX course from the Linux Foundation covers topics such as software-defined networking, automating tasks, and open networking operating systems.
▶ **magpi.cc/lfnetwork**

THE BITS AND BYTES OF COMPUTER NETWORKING
Ideal for beginners, this Google course on Coursera offers 27 hours of learning, covering modern networking fundamentals and troubleshooting techniques.
▶ **magpi.cc/bitsbytes**

SO YOU WANT TO BE A NETWORK ENGINEER?
The video lectures in this free Udemy course will give you a basic understanding of networking technologies and could set you on a new career path.
▶ **magpi.cc/soyouwant**

Ken
St. Cyr

The star of What's Ken Making on game emulation and his first experience of Raspberry Pi

> Name **Ken St. Cyr** | > Occupation **Software engineer**
> Community role **Video maker** | > URL **whatskenmaking.com**

Retro game emulation has been improving by leaps and bounds over the last decade through various technologies, and one of those is Raspberry Pi. It brings huge power in a small form factor and is perfect for the hobby, and game preservation.

Software engineer Ken St. Cyr runs a YouTube channel, called What's Ken Making, that's dedicated to modern retro gaming, which includes Raspberry Pi-powered devices.

"My earliest gaming memory is split between playing *Pitfall* on my uncle's Coleco Vision and typing up BASIC games from a book into a TI-99/4A," Ken tells us. "I also would go to my friend's house down the street to play his Atari VCS on a fuzzy black-and-white TV, which I'm pretty sure is the reason why I wear glasses today. After that, I received a NES for Christmas, and that's what I primarily played until I got a Tandy 1000HX and discovered Sierra adventure games. Even today, I still enjoy playing the *Police Quest*, *Kings Quest* and *Space Quest* series on my MiSTer's AO486 core!"

What is your history with making things?

I've been a maker for as long as I can remember. My interest in electronics started as a kid in the '80s when I got a Radio Shack electronics lab kit. It had these little springs, which you could attach wires between to make different circuits.

I spent hours wiring up the example circuits and trying out different things to see what would happen – it was really quite magical!

When did you learn about Raspberry Pi?

I want to say it was probably about 10 years ago now... I was teaching a class at work, and one of the students had a Raspberry Pi that he brought with him. I remember examining it after class one day and marvelling at it. I was tinkering with Arduino at the time, so the idea of a single-board computer that ran an operating system and gave me access to GPIO really enthralled me. I bought one as

▶ The hardware differences (and similarities) between the NES and its Japanese version the Famicom are fascinating

soon as I could and have been playing with them ever since.

What's your favourite video you've made?
That's like choosing a favourite child! If I had to pick one, I'd say it's probably the video series I'm working on right now, called 'Inside the Famicom', which is a multi-episode deep dive on Nintendo's Famicom and NES. I currently have nine

enjoy the teaching aspect of my videos more than anything, so the idea of parking on one topic for a period of time to unpack it is really appealing to me.

What have been some of your favourite projects?
My favourite projects tend to be microcontroller-based recreations, like the handheld ZX Spectrum that I made a video about earlier this year.

Heckendorn. We're working on version four now, and there are some interesting changes in the works that I think people will absolutely love! **M**

▲ The handheld made from a Raspberry Pi Zero when assembled

▼ Raspberry Pi Zero has always been great for slipping into smaller projects, like this handheld console

" My favourite projects tend to be microcontroller-based recreations, like the handheld ZX Spectrum "

episodes planned, which I'll be releasing throughout the summer – and the first episode went live last week. I really

I've also really enjoyed working on the gameBadge, a Pico-based handheld game system, with maker extraordinaire Ben

NASsie

A full-featured NAS solution using Raspberry Pi

We got an email from Jeff Loeliger about a very cool project he's been working on recently.

"When you have several computers and lots of files around the house it is useful to have a NAS – network attached storage – device to share files, stream media with Plex, and back things up," Jeff writes. "My current system is an old QNAP TS–251 and a very old, and unsupported, QNAP TS–110. I wanted something new and faster, which sounded like a project for the Raspberry Pi. It has a compact completely 3D–printed case with a custom pHAT interface board."

He's since gone on to write up a comprehensive tutorial on how to build a NASsie (**magpi.cc/nassieh**) and set up the software on it (**magpi.cc/nassies**). It uses a Raspberry Pi Compute Module 4 and runs OMV 6 (Open Media Vault) with custom software to run the little display interface on the case.

01. You can fit a lot of storage inside the case – both spinning platters and solid state

02. This really cool, Loch Ness Monster-inspired case, is 3D printed and very customisable

03. It's a tight squeeze, but that big case fan will keep everything cool

01

02
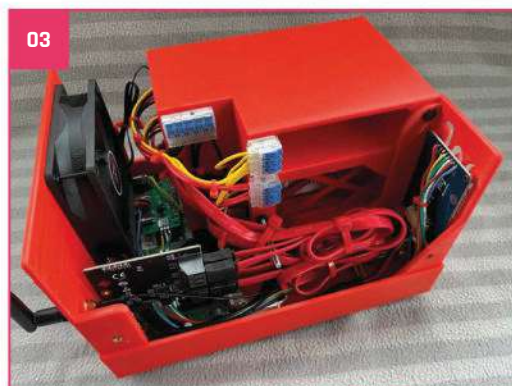
03

Events in pictures:
Roanoke Robotics Club Raspberry Jam

Community and official events in the wild
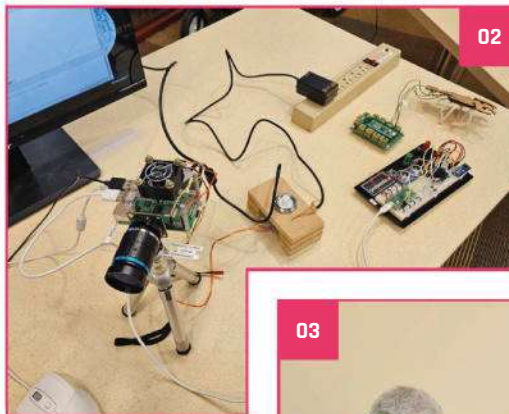
Did you know that if you hold a Raspberry Jam and get it listed on the events website, you'll be sent some swag for the event? Several months ago it included Raspberry Pi 5.

"The Roanoke Robotics Club held our Raspberry Pi Jam [on] Saturday March 2 at our neighbourhood library," Gary Yohe of the Roanoke Robotics Club tells us. "We had a great turnout and appreciate the Raspberry Pi 5s shipped just in time for the exhibits."

01. Roanoke Jam is held in a local library

02. Many projects were on display, including this Pico-controlled Raspberry Pi camera

03. Raspberry Pi 5 was put to the test against a Raspberry Pi 4

04. It's a whole operating system running off a Raspberry Pi Pico!

MagPi Monday

Amazing projects direct from social media!

Every Monday we ask the question: have you made something with a Raspberry Pi over the weekend. Every Monday, our followers send us amazing photos and videos of the things they've made.

Here's a selection of some of the awesome things we got sent this month. Remember to follow along at the hashtag #MagPiMonday!

01. Etching the case to show which switches have been thrown is a genius idea

02. We're always still learning ourselves! Fritzing is a great tool to have in your belt as well

03. The Pimoroni weather stations are some of our fave summer projects too – we did a list of them last issue

04. We can see where it gets its name – also a very good legacy

05. This musical project is getting there!

PenguinTutor (Stewart Watkiss)
@penguintutor@fosstodon.org

@themagpi
I designed a laser cut enclosure for my Raspberry Pi Pico based model railway point controller.

penguintutor.com/projects/lase...

01

kevinmcaleer • 2 h

This weekend I looked at 5 fun Raspberry Pi projects for Summer including a Weather Station project using the @pimoroni WeatherHat. It's one of my favourite ones because you can share your live data with Weather Underground and contribute to Citizen Data science! Link to video here youtube.com/live...

03

Pater Practicus
@paterpracticus@mastodon.world

@themagpi Learned a new skill today - my first #Fritzing schematic for the Pico level crossing light controller (youtu.be/lupTxBROI1U?si=Js50UB...).
#MagPiMonday

02

briancorteil 10/04/2024 **04**

Been putting the finishing touches to our PiWars robot Scorpion, build around the Tiny 4WD and the Raspberry Pi Zero 2. Scorpion is a direct descendant of the MagPi Tiny created 8 years ago!

diyelectromusic @diyelectromusic@mastodon.social

@themagpi We've merged in preliminary Raspberry Pi 5 support to MiniDexed thanks to Rene's excellent Circle bare metal OS now supporting it. It's still considered experimental and doesn't currently do anything a Pi 3 or 4 can't do, but it does appear to work.

The photo is me testing PWM audio output on my Pi 5.

#MiniDexed #RaspberryPi #MagPiMonday

github.com/probonopd/MiniDexed

05

Crowdfund this

Great crowdfunding projects this month

XGO-Rider

We've reviewed XGO robots in the past but none of them have looked like this two-wheeled bipedal robot. The videos on the Kickstarter page are very impressive as they follow people around with very little wobbling at all.

▶ **kck.st/3y6G6wW**

RaceBox Micro

This is a DIY GPS chip compatible with Raspberry Pi and includes an accelerometer and gyroscope. It's being marketed for hobbyists with remote control cars, drones, planes, etc, and comes with a huge range of features for tracking your device's movements.

▶ **kck.st/43ZsZtk**

Your
Letters

Future forecast

Very much enjoyed the 'Final Word' in [*The MagPi* 140]. I currently teach GCSE computer science, and specifically Python programming, to a generation of pupils that are encountering the dawn of the AI age. The past 18 months have made me question the value of what I am teaching them, and your point about abstraction is one that I have made to several of my colleagues.

Python already abstracts machine code, so why should we object to GPT abstracting writing the Python? Should I be teaching pupils how to phrase questions in a way that will generate the most efficient code from GPT? It's not hard to foresee a time when we can ask GPT to cook up a recipe, code, SLT drawing files and parts list to create some of the projects that are typically shown in your magazine.

Luckily I am closer to retirement than the start of my career; I have this nagging feeling that things are going to change significantly and quicker than people realise.

Paul via email

With stuff like code it's very easy to see how the LLMs would be able to put together great code for people to tweak thanks to a large open-source dataset with instructions and such – we've seen a couple of ML-generated STLs, and they might be a bit further off though. Hopefully the power and computation requirements get more efficient in the future for it!
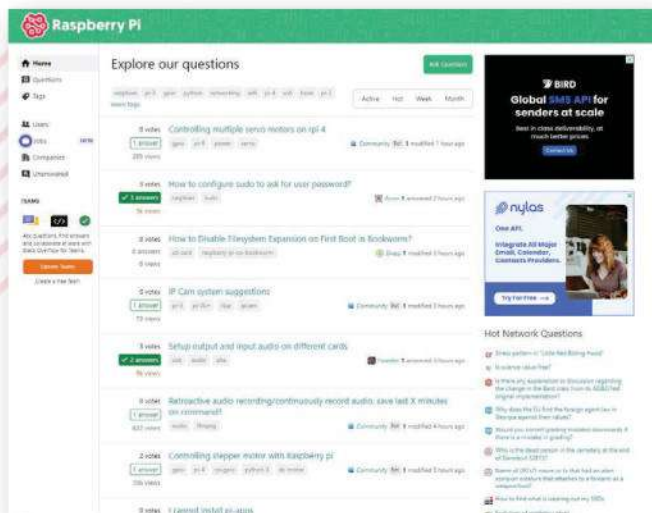
▶ We don't think Stack Exchange is going anywhere just yet though

I'm new to the world of Raspberry Pi. I recently purchased a Raspberry Pi 5 for a film digitiser project. My project that involves a bit of I/O using interrupts with an encoder, stepper motor control and a few other general switches. I'm looking for resources for the GPIO. It seems the new RP1 chip has thrown the development community a new challenge.

What resources does *The MagPi* team suggest for the Raspberry Pi5 GPIO tasks?

Steven via email

The Raspberry Pi docs (**rptl.io/docs**) are always a great start for this – they're very comprehensive, and have examples for just about every function for Raspberry Pi in Raspberry Pi OS. The Raspberry Pi forums (**magpi.cc/forums**) are also your number one place for more info if something is eluding you.
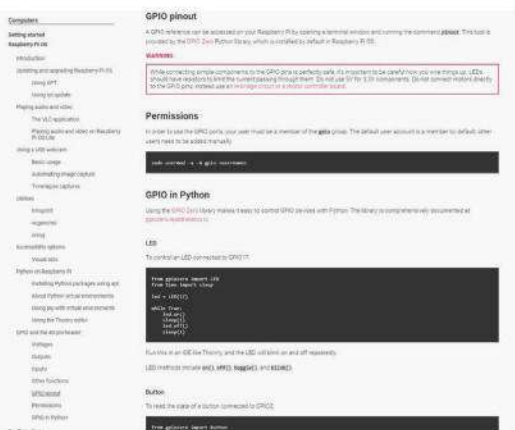
▲ The Raspberry Pi documentation is really, really comprehensive. We're often surprised ourselves!

Contact us!

➤ Mastodon	**magpi.cc/mastodon**
➤ Threads	**@themagpimag**
➤ Facebook	**magpi.cc/facebook**
➤ Email	**magpi@raspberrypi.com**
➤ Online	**forums.raspberrypi.com**

Community
Events Calendar

Find out what community-organised Raspberry Pi-themed events are happening near you...

01. Melbourne Raspberry Pi Meetup
- 🗓 **Sunday 2 June**
- 📍 **107 Victoria Harbour Promenade, Melbourne, Australia**
- ▶ **magpi.cc/mrpm142**

Open to everyone with an interest in electronics, robotics, home automation, 3D printing, laser cutting, amateur radio, high altitude balloons, space tech, etc. Makers are invited to bring along projects and project ideas, and come connect with other makers. Get your questions answered, show off your work, and get support to resolve nagging issues.

02. Code Battle
- 🗓 **Saturday 8 June and Sunday 9 June**
- 📍 **Campus Idéale, Nabeul, Tunisia**
- ▶ **magpi.cc/cb142**

The Code Battle competition around water resources with ScratchJr, Scratch, micro:bit and Raspberry Pi Pico aims to raise awareness among kids of the importance of sustainable water management while developing their programming and technology skills.

03. Cornwall Tech Jam
- 🗓 **Saturday 8 June**
- 📍 **Fraddon Village Hall, Saint Columb, UK**
- ▶ **magpi.cc/ctj142**

Cornwall Tech Jams are run by volunteers working in IT and education throughout Cornwall. They are supported by Software Cornwall, its members and other local businesses. Our volunteers give their own time and expertise to plan each Cornwall Tech Jam and to the regular maintenance of all our equipment.

04. Riverside Raspberry Pi Meetup
- 🗓 **Monday 10 June**
- 📍 **3600 Lime Street, Riverside, CA, USA**
- ▶ **magpi.cc/rrpm142**

The purpose of Riverside Raspberry is to share knowledge related to Raspberry Pi hardware in particular, and to promote interest in tech development in the Inland Empire in general. The group is currently meeting on the second Monday evening of every month.

FULL CALENDAR
Get a full list of upcoming community events here:
magpi.cc/events

INTERNET OF THINGS WITH RASPBERRY PI PICO
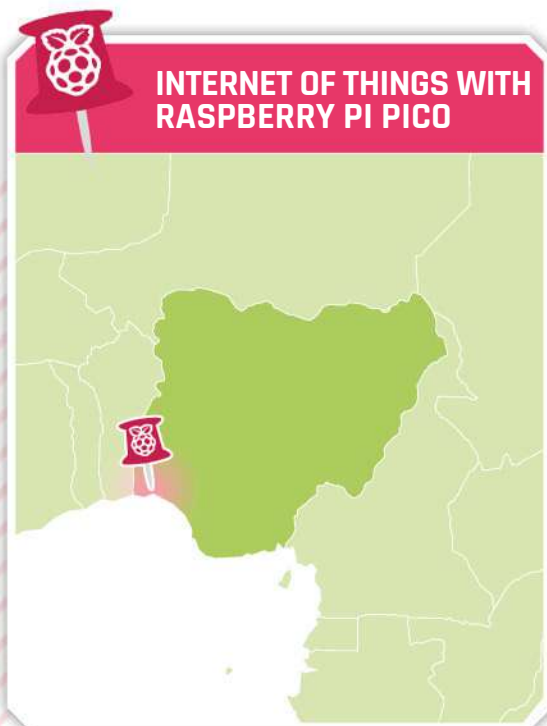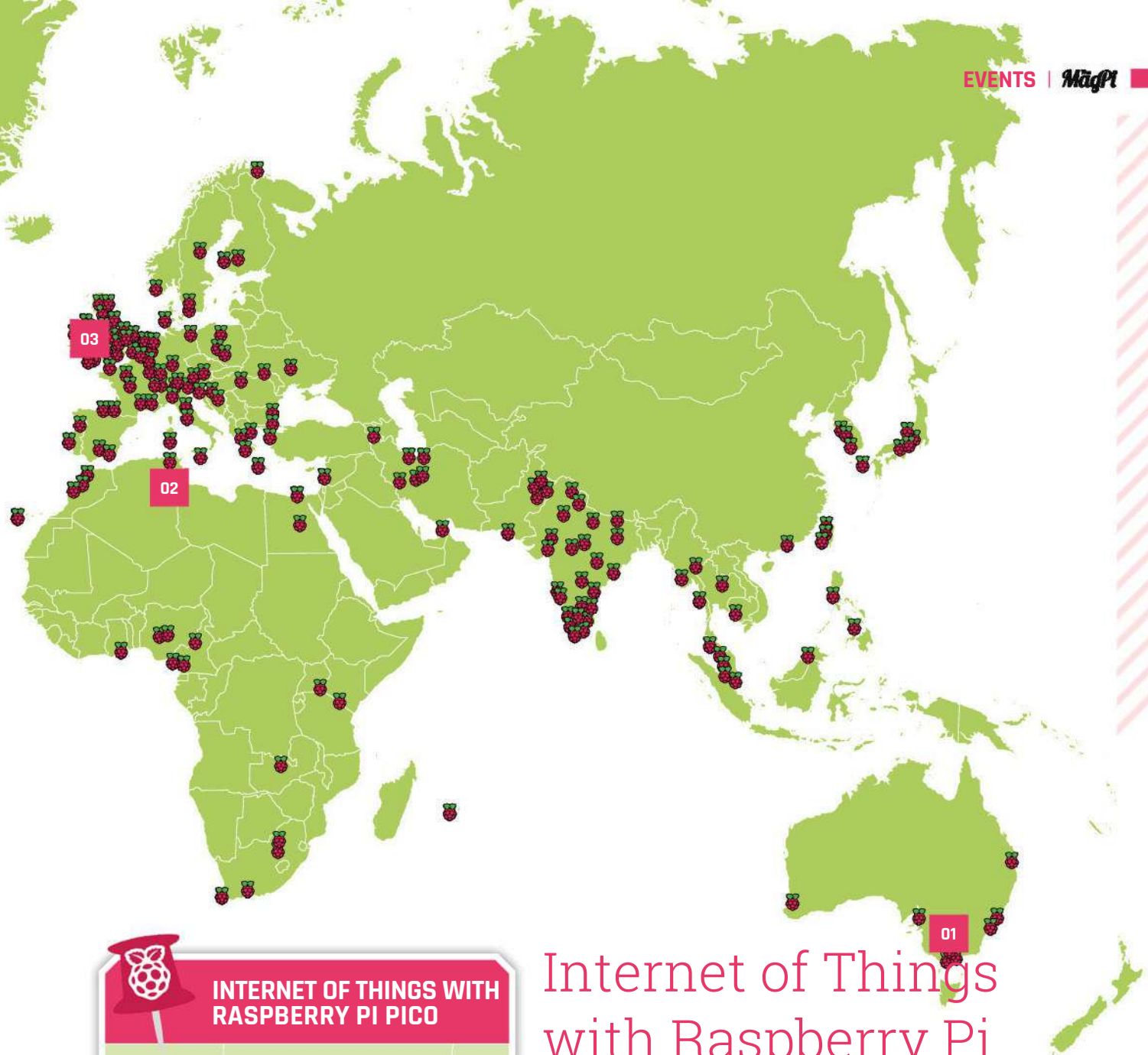
Internet of Things with Raspberry Pi Pico workshop

> Where   Lagos, Nigeria
> When   Monday 3 June to Thursday 6 June

The Internet of Things with Raspberry Pi Pico is a free workshop taking place in Lagos, Nigeria. Over the course of two days, participants will learn how to use a Raspberry Pi Pico microcontroller to solve real-world challenges using sensors, networking, and cloud technology. This workshop is intended for enthusiasts and university students in the Lagos technology community and it's hosted by Raspberry Pi.

magpi.cc/iotpico24

WIN ONE OF TEN
M.2 HAT+

Unleash the power of PCIe with the release of the much anticipated M.2 HAT+ – allowing for M.2 SSD cards, machine learning, and many more high-speed operations. We have ten to give away to lucky readers.

Head here to enter: magpi.cc/win | **Learn more: magpi.cc/m2hat**

Terms & Conditions

Raspberry Pi Custom DSI Display
CM4 Carryboard, and Industrial Tablet
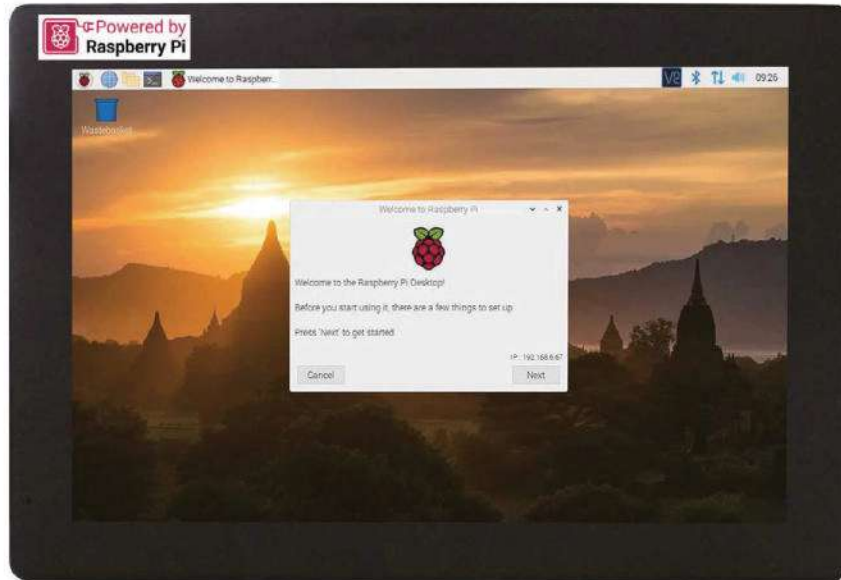
Custom DSI Displays
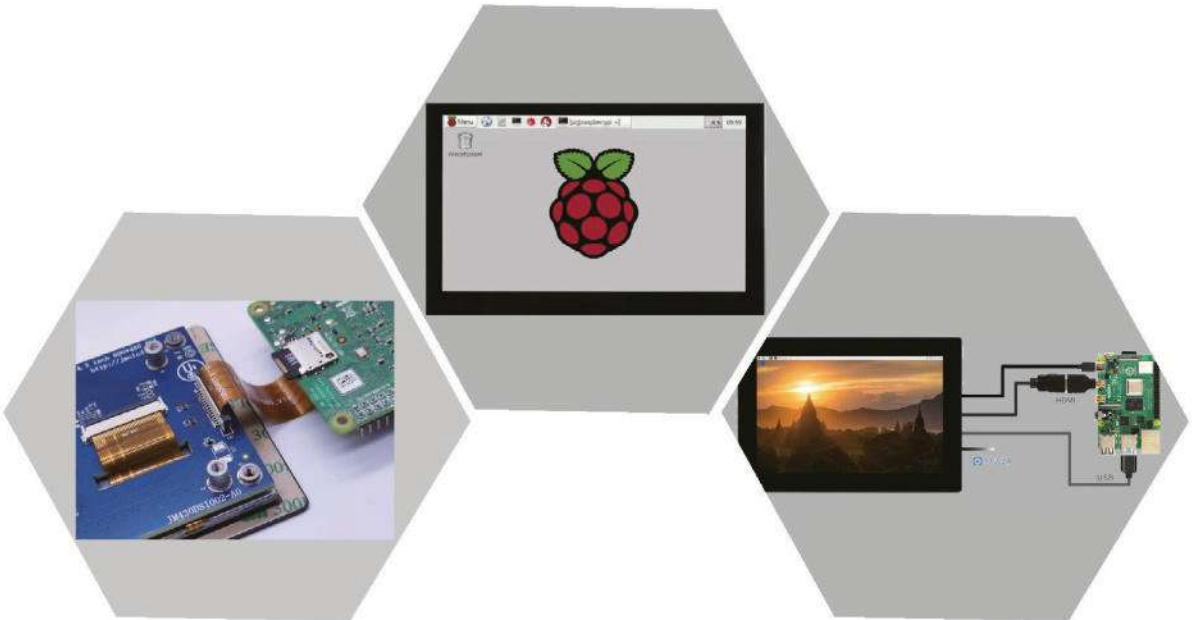
Development of
CM4-compatible carryboard

Industrial Tablet
development capabilities

Looking for cooperation partners around the world.

Contact us
Tel: +44 7587611960
E-mail: straight@jmolcd.com
Website: www.jmolcd.com

STARTER ROBOTICS

BUILD WHEELED, WALKING, AND AI-POWERED ROBOTS
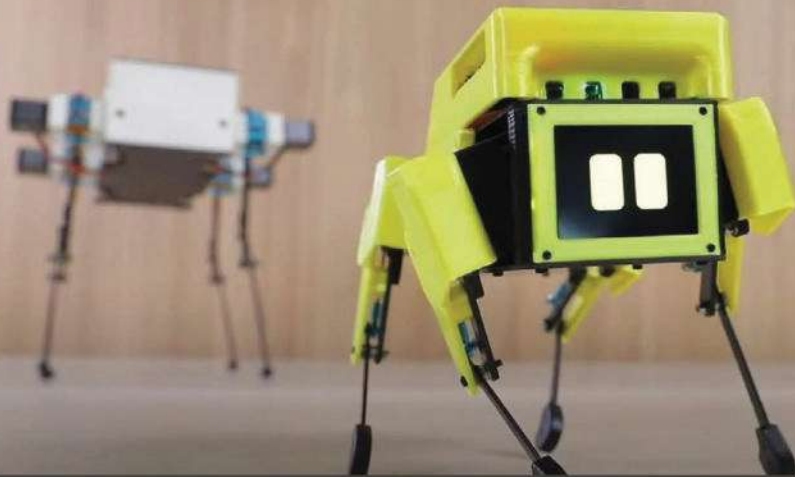
The Magpi **#143**
On sale **27 June**

Plus!

Raspberry Pi
in industry

**A giant
retro mirror**

The
Dicemaster 2000

DON'T MISS OUT! magpi.cc/subscribe

MASTODON	magpi.cc/mastodon
THREADS	@themagpimag
FACEBOOK	magpi.cc/facebook
EMAIL	magpi@raspberrypi.com
ONLINE	forums.raspberrypi.com

Summer time

The sun is starting to shine which causes a dilemma for homebody **Rob Zwetsloot**

I n the (fantastic) game *Stardew Valley*, you play out several years each split up into four distinct seasons lasting 28 days. It's a farming sim after all. On the last day of summer there's a nighttime event that ends with a load of glowing jellyfish swimming by before a message pop ups: "the glow of summer has faded, now… and the moonlight jellies carry on toward the great unknown".

I've never had any real melancholy for summer ending – aside from, I suppose, having to go back to school

> **That made me realise that a long exposure Raspberry Pi camera would have worked perfectly… just like in the many astrophotography features I'd written**

when I was a kid – but while reading the game's text I think I came close to understanding why people do.

I do like summer though; after months of cold and dreary rain, it's lovely to have a bit of sun. It also helps that I live near the beach so I can go for sunny and beautiful walks. I'm also pretty good with a barbecue if I do say so myself.

At the time of writing this Final Word, the aurora borealis had appeared all over the UK thanks to Earth being in the path of a solar storm and… I missed it. I had no idea it was happening until I was snuggled up in bed on a Friday night and friends from Scotland and northern England started sharing photos. Good for them, I thought! It wasn't until the morning I found out it had also reached my end of the country.

Suit your needs

I went out the following night to try and catch a glimpse but unfortunately for me – and the hoard of locals convening at a dark corner of the beach – there was nothing to see apart from some pretty stars.

People had been taking photos on smartphones with night shot features, and that made me realise that a long-exposure Raspberry Pi camera would have worked perfectly… just like in the many astrophotography features I'd written. Unfortunately it was too late for photography that night, but it got me thinking about other outdoor activities I do which could be

improved with a Raspberry Pi, rather than forcing myself to do something new just to use the SBC.

Last year I decided on a near-whim to go hiking in the Peak District with some friends. It was a very wet August weekend so I'm glad I didn't take a Raspberry Pi with me, but I understand there is a ton of geocaching you can do in the area. I've never played around with a geocache project, so if I revive the concept this year I might just have to make a Raspberry Pi-powered one.

Staying in

Unfortunately I also like staying home a lot. Maybe I need to invent a Raspberry Pi-powered air conditioner for myself as it is getting extremely roasty-toasty at home these days. I'm also always very interested in some minor gardening, whether it be herbs or some flowers, so perhaps this is the year I finally make an automated plant watering system.

Well, with all that planning out of the way, maybe I can start thinking about Autumnal projects. Those long months of sunshine and heat can get oppressive… **M**

AUTHOR

Rob Zwetsloot

Rob is not really much of an amateur astronomer but looking up at the stars at 1am when it's a pleasant 18°c while the waves are lapping nearby sounds nice

magpi.cc